Video Compression Algorithms using Machine Learning Approach for Encoding Time Reduction in HEVC

Submitted in partial fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy

by

Sanagavarapu Karthik Sairam
(Roll No: 718041)

Under the supervision of

Dr. P. Muralidhar
(Associate Professor)

Department of Electronics & Communication Engineering

National Institute of Technology Warangal

Telangana, India - 506004

2022

Dedicated

To

My Family,

Teachers & Friends

Approval Sheet

This thesis entitled **Video Compression Algorithms using Machine Learning Approach for Encoding Time Reduction in HEVC** by **Sanagavarapu Karthik Sairam** is approved for the degree of **Doctor of Philosophy**.

Examiners

Research Supervisor

Dr. P. Muralidhar
Department of ECE
NIT Warangal, India-506004

Chairman & Head

Prof. P. Sreehari Rao
Department of ECE
NIT Warangal, India-506004

Place:

Date:

Declaration

This is to certify that the work presented in this thesis entitled **Video Compression Algorithms using Machine Learning Approach for Encoding Time Reduction in HEVC** is a bonafied work done by me under the supervision of **Dr. P. Muralidhar** and was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my own ideas and even considered others ideas which are adequately cited and further referenced the original sources. I understand that any violation of the above will cause disciplinary action by the institute and can also evoke panel action from the sources or from whom proper permission has not been taken when needed. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea or data or fact or source in my submission.

Sanagavarapu Karthik Sairam
Research Scholar
Roll No.: 718041

Place:
Date:

NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL, INDIA-506004

Department of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the thesis work entitled **Video Compression Algorithms using Machine Learning Approach for Encoding Time Reduction in HEVC** is a bonafide record of work carried out by **Sanagavarapu Karthik Sairam** submitted to the faculty of **Electronics & Communication Engineering** department, in partial fulfilment of the requirements for the award of the degree of **Doctor of Philosophy** in **Electronics and Communication Engineering, National Institute of Technology Warangal, India-506004**. The contributions embodied in this thesis have not been submitted to any other university or institute for the award of any degree.

Dr. P. Muralidhar
Research Supervisor
Associate Professor
Department of ECE
NIT Warangal, India-506 004.

Place:

Date:

Acknowledgements

First, I take immense pleasure to convey my sincere gratitude to my supervisor Dr. P. Muralidhar for his perpetual encouragement and supervision. His guidance has oriented me in a proper direction and supported me with promptness and care.

I am also grateful to Prof. P. Sreehari Rao, Head, Department of ECE, for his invaluable assistance and suggestions that he shared during my research tenure.

I take this privilege to thank all my Doctoral Scrutiny Committee members Prof. C. B. Rama Rao, Prof. P. Sreehari Rao, and Prof. J. V. Ramana Murthy (Dept. of Mathematics) for their detailed review, constructive suggestions, and excellent advice during the progress of this research work. I would like to extend my thanks to Prof. T. Kishore Kumar, Prof. N. Bheema Rao and Prof. L. Anjaneyulu for the insightful comments and encouragement. I would also like to thank all the faculty of Dept. of ECE who helped me during the course.

I would like to convey my sincere gratitute to P. Raveendra, J. Nandini, Y. Rama Muni Reddy, J. Ashish, Dr. K. Krishna Reddy and C. Jayaram for thier invaluable guidance and support. I would also like to extend my heartfelt appreciation to my family, colleague scholars, friends, and well-wishers who helped to write my thesis with their support. Finally, I thank my nation India, for giving me the opportunity to carry out my research work at the NIT Warangal. A special thanks to MHRD for its financial support.

Sanagavarapu Karthik Sairam

Abstract

High-definition (HD) and ultra HD videos have recently become a prerequisite for applications such as security cameras, television systems, etc. However, the density of visual data grows dramatically due to the increase in video resolution, making it difficult to store, transmit, and process HD video data. This leads to the development of the High Efficiency Video Coding (HEVC) standard. The HEVC compresses video sequences with 50% less bitrate than the H.264 standard. The high efficiency in HEVC is achieved mainly due to the advanced quad-tree structure and motion estimation. Motion estimation helps to determine the motion vectors, and the quad-tree structure is obtained by recursively splitting the largest Coding Tree Unit (CTU) until it reaches the smallest size.

The motion estimation process increases the encoding time. In addition, the search pattern used in the motion estimation process may get trapped to local minima resulting in inaccurate motion vectors. Moreover, HEVC uses Rate-Distortion Optimization (RDO) search process to determine the optimal partitions in the CTU. The CTU quad-tree structure improves efficiency at the cost of high computational complexity. The complexity is due to splitting 64×64 CTU into 85 coding units at four depths, namely depths 0, 1, 2, and 3, and verifying the Rate-Distortion cost of each CU from bottom to top. This process increases the encoding time. This thesis concentrates on reducing the encoding time of the encoder by accelerating the motion estimation process and predicting the optimal CTU partitions using a machine learning approach.

In this thesis, the first work concentrates on reducing the computational complexity of the motion estimation process using the Multi-Level Resolution Vertical Subsampling (MLRVS) algorithm. The MLRVS algorithm is a motion estimation approach that helps to accelerate the motion vector calculation process. This algorithm uses three different resolution frames, which are created using the Vertical Subsampling process. Vertical Subsampling minimizes the Sum of Absolute Difference (SAD) computations by considering only even rows in the frame. The search operation using search patterns is performed

in three frames, starting at the lowest resolution frame and ending at the original frame, to determine the global minima. The complexity reduction algorithm is also proposed to determine the skip mode early, which decreases the encoding time. Two distinct search patterns are suggested to reduce the time needed to locate the motion vectors.

In addition to the motion estimation algorithm, the second work emphasizes reducing the complexity of the conventional RDO search process by using a Convolutional Neural Network (CNN). The CNN is trained using the CU depth data derived from the HM reference software to predict the Coding Unit (CU) size. Supervised learning is performed by giving the luminance CTU as an input and the CU depth as the output. Based on the output (0 or 1), the decision to split the CU is chosen during the testing operation. After determining the CU size, the motion estimation is performed for each partition using the Multi-Resolution frame with the Cross Diamond Octagonal search pattern (MRCDO). Besides, the CDO search pattern is used to speed up the motion vector calculation.

The CNN targets the spatial features of the image. The accuracy of the model for the video data can be increased using spatial and temporal features. Hence, the third work focus on reducing the computational complexity in Scalable HEVC (SHVC) by using the Early Terminated CNN (ET-CNN) and Early Terminated Long- and Short-Term Memory (ET-LSTM) approach. SHVC is the extension of HEVC that allows the same video encoding into various video resolutions, qualities, or frame rates. We designed the ET-LSTM network that predicts the CU partition by taking the output features of the ET-CNN. Using the CU depth training data, the ET-CNN learns the CU partitions from the residual Coding Tree Unit (CTU). In addition, Horizontal Subsampling Motion Estimation (HSME) method is proposed to find accurate motion vectors with reduced complexity.

The fourth work utilizes the previous work scalable HEVC using ET-CNN and ET-LSTM to encode the Region of Interest (ROI) for surgical telementoring application. The surgical telementoring system transmits the video with high quality and less bit rate. The transmission of high-quality video with less bitrate is very difficult to achieve. Hence, the Region of Interest (ROI), i.e., the surgical incision region in the frame, is transmitted with high quality, and the rest of the frame is transmitted with very low quality. We proposed segmenting the surgical incision region using the Kernelized Correlation Filter (KCF) object tracking technique. The KCF tracker uses a correlation filter for training,

detection, and ROI tracking. Then the complexity-efficient SHVC encodes the segmented region to meet the resolution of an end-user device.

The last work deals with the region-based motion estimation using YOLOv4 and the next frame prediction using the Neural Network. Initially, YOLOv4 is trained using the labeled data of the frame. The labeled data represents the different regions in the frame. After training, the bounding box of each region in the test video sequence frames is detected using YOLOv4. Then the bounding box coordinates of each region are derived, and the centroid point of each bounding box is calculated. The motion vector of each region in the current frame is given using the centroid point of the corresponding region in the previous frame. This approach considerably minimizes the computational complexity of the HEVC motion estimation process.

During the next frame prediction process, the neural network is trained with the first five frames of the sequence as an input and the sixth frame as the output. The neural network uses layers such as Reshape, Permute, Lambda, Gaussian Noise, Upsampling2D, and ConvLSTM2D layers to predict the sixth frame. The images of size 96×96 with batch size 32 are given as input. Adam optimizer and the Mean Square Error loss function are used during training. The perceptual distance is used to determine the similarity between the frames. The results show that the proposed approach can efficiently predict the sixth frame.

The proposed approaches in this thesis significantly decrease the encoding time with minor degradation in quality.

Contents

List of Figures

List of Tables

List of Abbreviations

HEVC	High Efficiency Video Coding
AVC	Advanced Video Coding
CU	Coding Unit
CTU	Coding Tree Unit
QP	Quantization Parameter
MV	Motion Vector
SAO	Sample Adaptive Offset
CTB	Coding Tree Block
CB	Coding Block
PB	Prediction Block
ME	Motion Estimation
MC	Motion Compensation
SW	Search Window
PU	Prediction Unit
TB	Transform Block
RDO	Rate-Distortion Optimization
RD	Rate-Distortion
SAD	Sum of Absolute Difference
ROI	Region of Interest
MVP	Motion Vector Predictor
CBF	Coded Block Flag
SR	Search Range
KCF	Kernelized Correlation Filter
GOP	Group of Pictures

Chapter 1

Introduction

1.1 Introduction

The advent of new technologies has resulted in significant growth in the consumption of multimedia information in recent years. The video content now accounts for nearly half of all web traffic. The giant companies such as Netflix, Amazon, etc., are using large amounts of data to entertain subscribers with high-quality video content. The statistics in Fig. 1.1 exhibit that the consumers are seeking higher-quality content for improved visual experience. The high quality video content increases the bandwidth required to transmit the data through the channel. Thus, video coding standards are increasingly important for maintaining present coding efficiency and providing the necessary encoding tools to accommodate new formats such as ultra-high definition (UHD) video sequences.

The Joint Collaborative Team on Video Coding (JCT-VC) created the video codec known as the High Efficiency Video Coding (HEVC) [2] that can support UHD videos. This standard seeks to increase encoding performance by reducing the bitrate while maintaining the same visual quality. The metrics in Table 1.1 show that HEVC achieves approximately 50% bitrate reduction compared to its forerunner, the H.264 Advanced Video Coding (AVC) standard [3]. HEVC improves coding efficiency using advanced tools like Quad-tree Coding Tree Unit (CTU) structure, intra prediction, inter prediction, and in-loop filters. These increased capabilities have piqued the interest of industry leaders, with many already incorporating the standard into a wide range of devices. This gain in compression efficiency comes at the expense of computational complexity, which is the

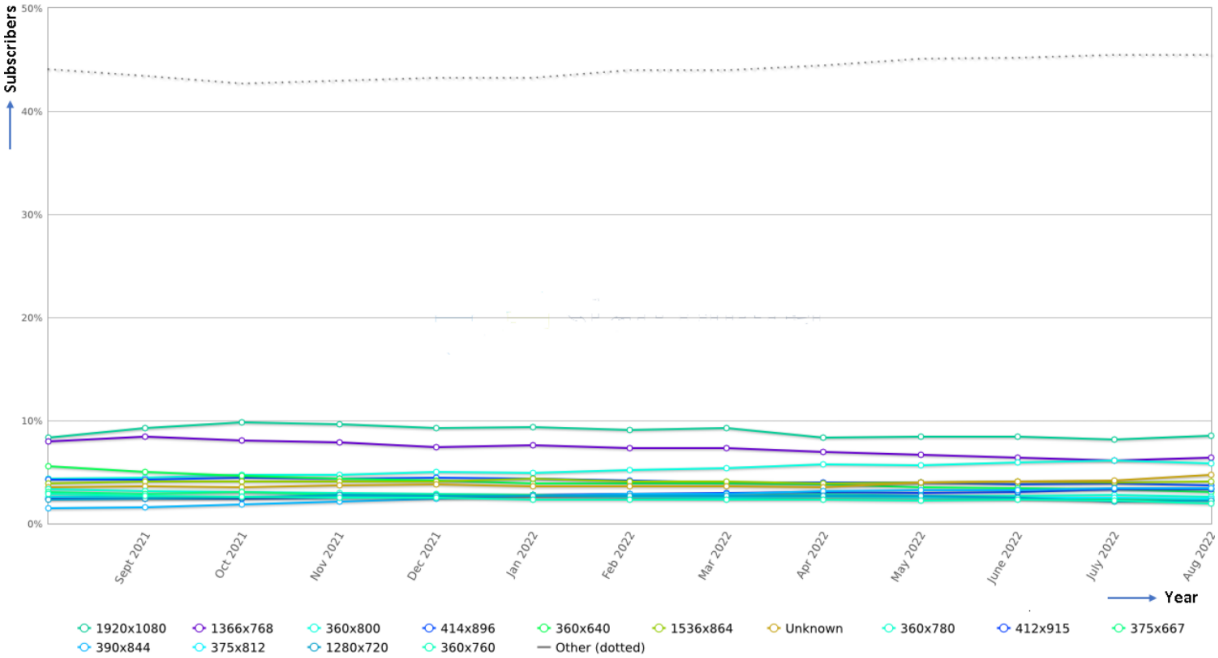


Figure 1.1 Statistics of various video resolution subscribers [1]

biggest impediment to HEVC inclusion in real-time applications.

Table 1.1 Comparison of subjective video performance between HEVC and H.264/AVC [4]

Video codec	**Average bitrate reduction compared to H.264/AVC High Profile**			
	2160p	**1080p**	**720p**	**480p**
HEVC	64%	62%	56%	52%

1.2 HEVC standard

HEVC is a block-based video coding standard, and its coding mechanism is remarkably similar to that of its ancestor, H.264/AVC. However, HEVC substitutes the Coding Tree Unit (CTU) as a new processing unit for the macro-block structure. A simplified block diagram of the HEVC is shown in Fig. 1.2.

A typical encoding algorithm that produces an HEVC-compatible bitstream works as follows. Each image is divided into block-shaped areas, with the decoder informed of the precise block partitioning. Intra prediction is applied for the first frame of the

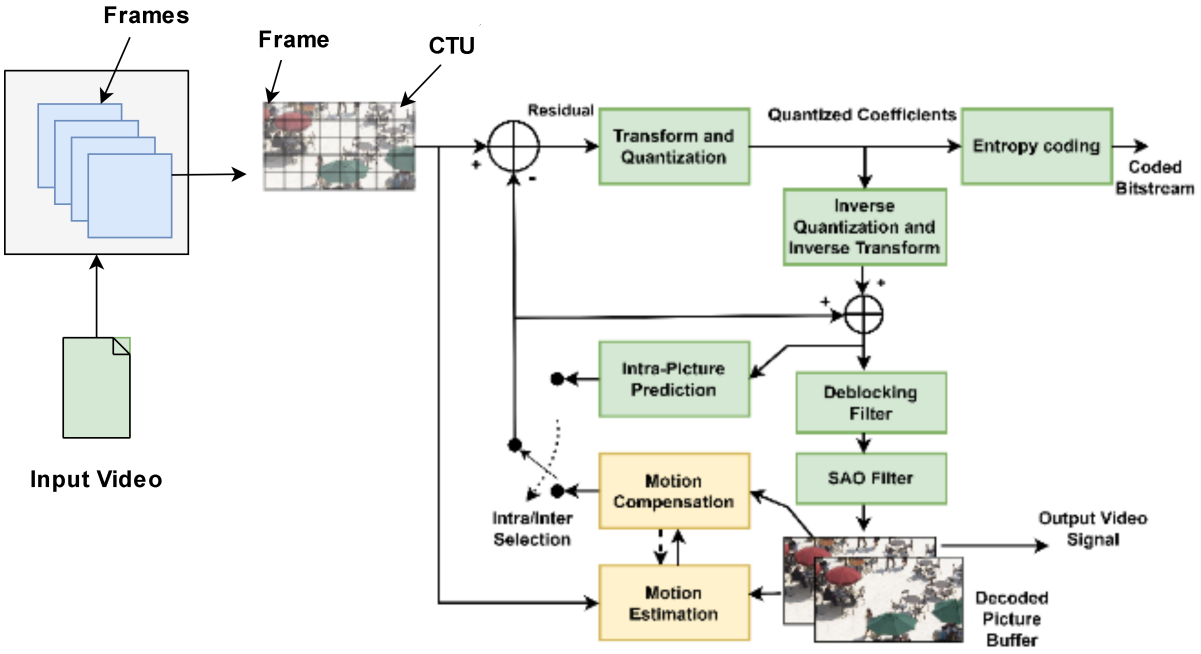


Figure 1.2 Block diagram of HEVC

sequence, and the rest of the frames use inter prediction in the sequence or between the random-access points. The inter picture encoding process involves choosing the motion data that consists of the selected reference picture and the Motion Vector (MV) data, which needs to be applied to predict every block's samples. The linear spatial transform is applied to the residual block, which is the difference between the current and predicted block. The resultant is quantized and then encoded by the entropy encoder.

The quantized coefficients are inverse transformed during the decoding process to generate the approximate residual block. The residual block is added to the prediction and then passed through the Deblocking and Sample Adaptive Offset (SAO) filters to remove the visual artifacts that occur at the boundaries of the block and for the smooth reconstruction of the output picture. The resultant output is stored in the Decoded Picture Buffer.

The detailed explanation of each block in the Fig. 1.2 is given below.

1.2.1 Coding Tree Unit (CTU)

The video sequence is the sequence of frames, and each frame is partitioned into square blocks called CTUs. Each CTU consists of a luma Coding Tree Block (CTB), associated chroma CTBs and syntax components. The size L×L of the CTU can be set to L = 16, 32, or 64 samples, with the bigger size often allowing for better compression. The CTU can be broken down into chunks called Coding Units using the quadtree-like structure and quadtree signaling. The CTU is connected to the quadtree's root. A CTU may have a single CU or may be divided into several CUs, and each CU has a segmentation into Prediction Units (PUs), and a tree of Transform Units (TUs) linked with it as shown in Fig. 1.3. A coding unit (CU) comprises one luma Coding Block (CB), typically two chroma CBs, and associated syntax.

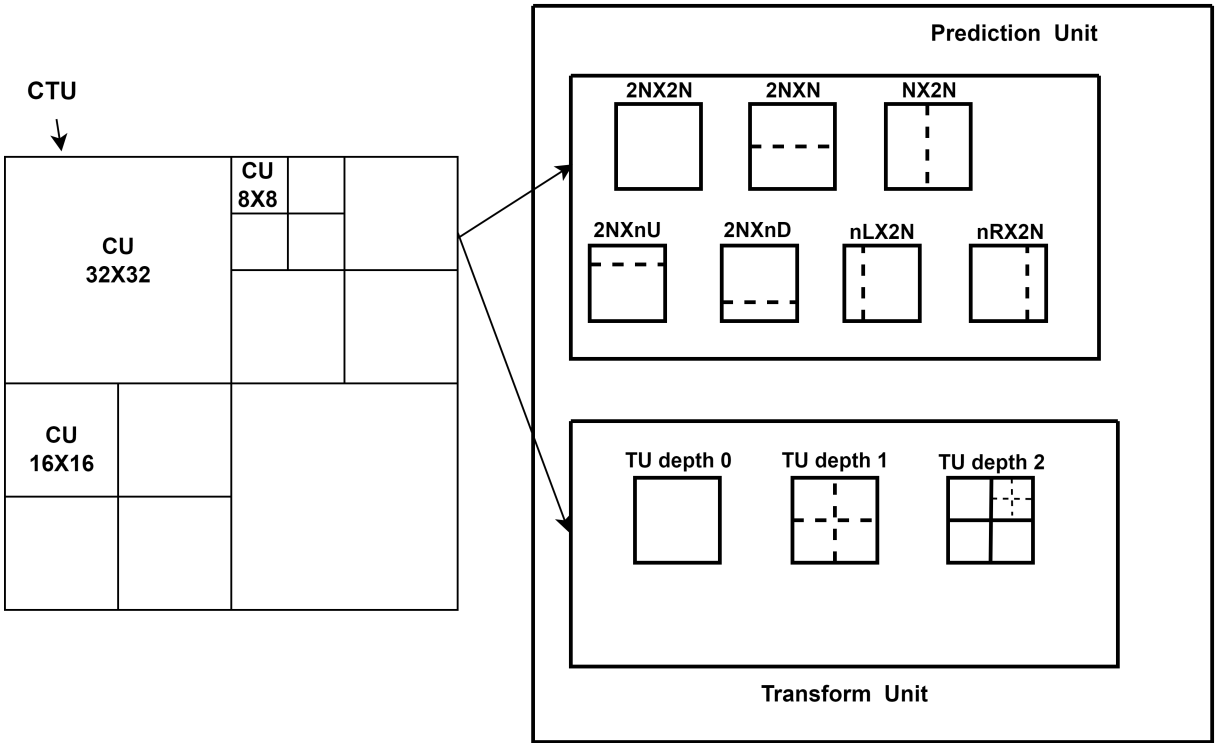


Figure 1.3 Coding Unit segmentation structure

1.2.2 Prediction Unit (PU)

The decision on coding the image area using inter or intra prediction is decided at the CU level. The root of the PU partitioning structure is at the level of CU. Based on the

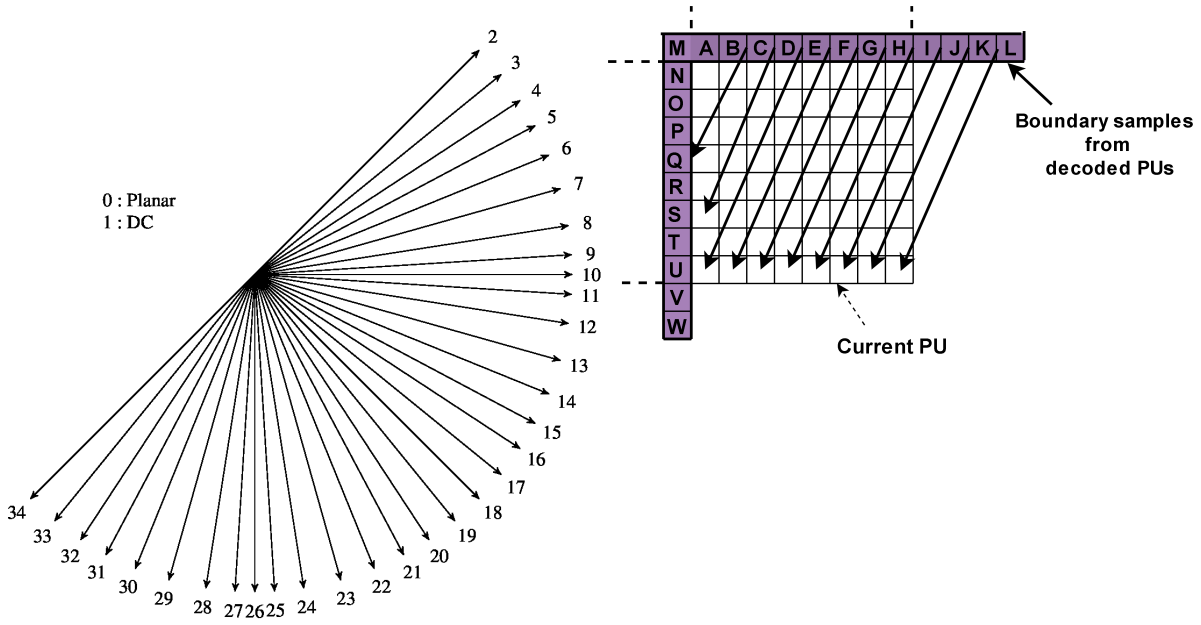


Figure 1.4 Pictorial representation of (a) Intra Prediction modes (b) Directional mode 29 in HEVC

type of prediction chosen at the CU level, the CUs are further split into different sizes and predicted from the luma and Chroma Prediction Blocks (PB). HEVC supports different sizes of PBs ranging from 64×64 to 4×4 samples. The Intra-Picture and Inter-Picture prediction are explained below.

1. **Intra-Picture Prediction:** It is used to remove spatial redundancy. The intra Picture prediction in the region is performed by considering the already decoded samples of the adjacent blocks as a reference. HEVC supports 33 directional, one planar, and one DC prediction mode. Fig. 1.4 (a) shows the 35 intra prediction modes and Fig. 1.4 (b) depicts the example representation of Intra prediction Directional mode 29. In directional mode encoding, the prediction is made using the spatially adjacent decoded blocks as a reference and the chosen angle to cover the current prediction unit. This method is ideal for places with strong directional edges. All the block sizes and prediction directions support directional mode prediction. The DC mode encoding merely utilizes a single value that correlates to the mean value of the boundary samples for the prediction.

2. **Inter Prediction:** A video is described as a collection of frames showing a continuous scenario in which each frame has a strong relationship to the frames that come after it. The moving objects change their position while the rest of the objects in the frames remain stationary. The Interpicture prediction in video coding helps to attain a high compression ratio since the prediction depends on the earlier coded frames. It uses two steps: Motion Estimation (ME) and Motion Compensation (MC). Motion Estimation (ME) is the process of determining the motion vectors that help to predict the current frame from the previous or reference frame. Based on the motion vectors, the contents of the blocks in the reference frame are moved for a more accurate prediction of the current frame. This mechanism is called Motion Compensation (MC). The resultant MC frame is subtracted from the current frame to generate the Residual frame. It contains the details about the frame where the changes occur.

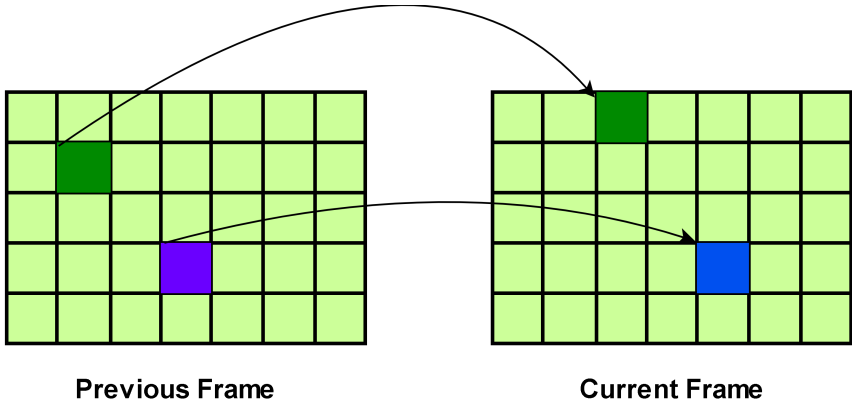


Figure 1.5 Uni-directional prediction

The prediction can be uni-directional or bi-directional based on the reference frame as shown in Fig. 1.5 and Fig. 1.6. The current frame is predicted from the previous frame in uni-directional prediction, previous (or earlier) and future frames in bi-directional prediction. The motion estimation is explained below.

(a) **Motion Estimation:**

Initially, the ME block takes the current block and the reference frame as input. Then the region of interest to perform the search operation is identified. The region of interest is called a Search Window (SW). After finding the SW, the

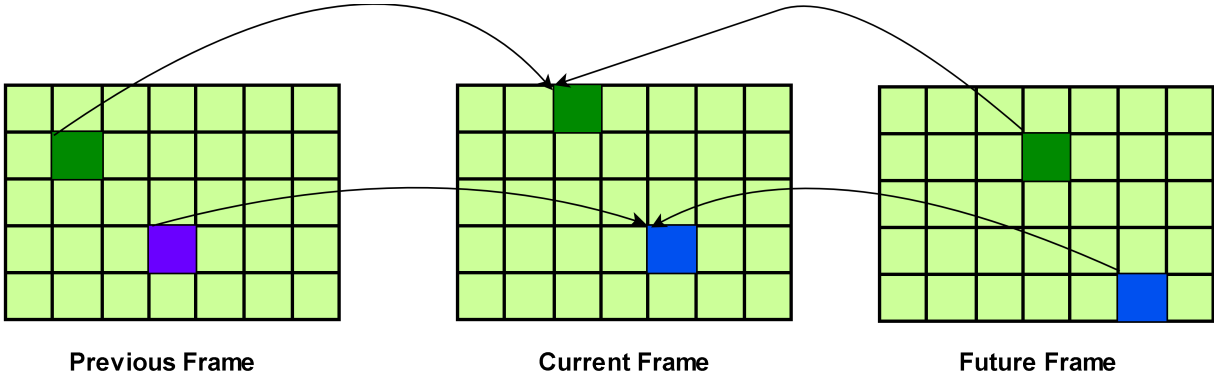


Figure 1.6 Bi-directional prediction

searching operation is performed based on the ME algorithm. The ME process is shown in Fig. 1.7. The searching can be done by using the Full Search (FS) algorithm; however, it gives accurate MVs at the cost of high computational complexity. Different fast search algorithms such as Diamond Search, Three Step Search (TSS), Hexagonal Search (HS), etc., are proposed to reduce the search points that help determine the MV quickly.

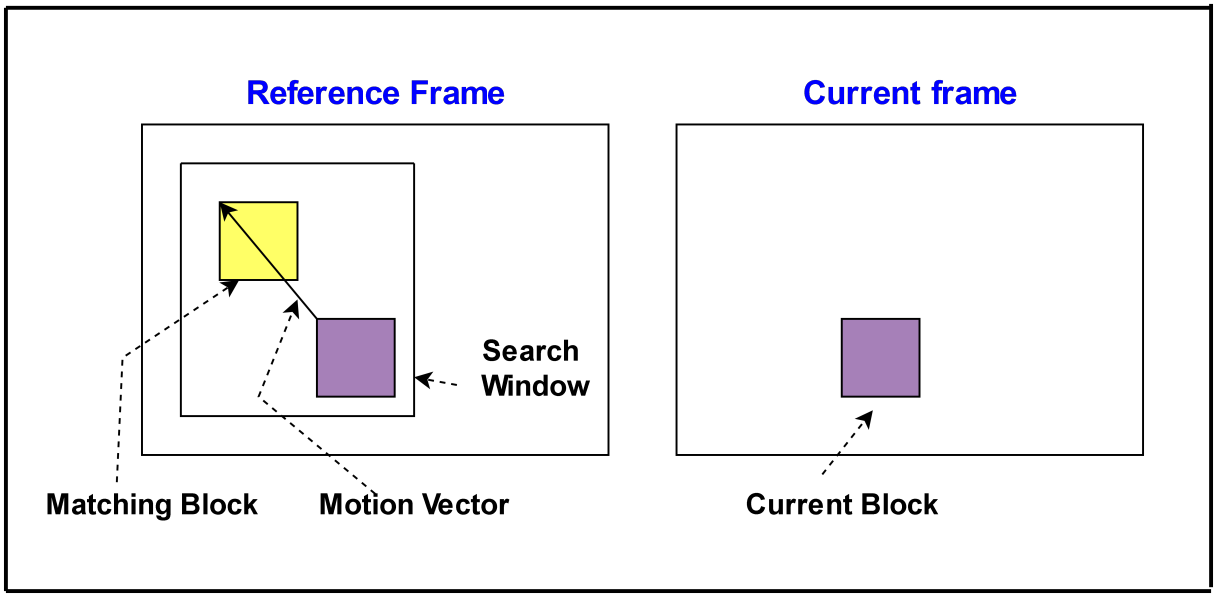


Figure 1.7 Block matching motion estimation

Sometimes the location of the prediction in the reference frame may be outside of the sampling grid where the intensity is uncertain. Hence the interpolation needs to be performed to determine the intensity of the positions between the

integer pixels. HEVC uses an 8-tap filter for luma and a 4-tap filter for chroma sample interpolation.

(b) **Variable Block Size Motion Estimation (VBSME):**

In HEVC, the motion estimation is carried out in various dimensions, one of which is the variance in block size, to obtain improved compression efficiency. To do this, the current block is divided into smaller sub-blocks, and ME is run for each sub-block. This process is known as Variable Block Size ME (VBSME). Then the output obtained after ME for the current block will be the MV for all the sub-blocks of the current block. However, all the sub-block sizes of the current block are considered, and the best mode is chosen for predicting the MC frame.

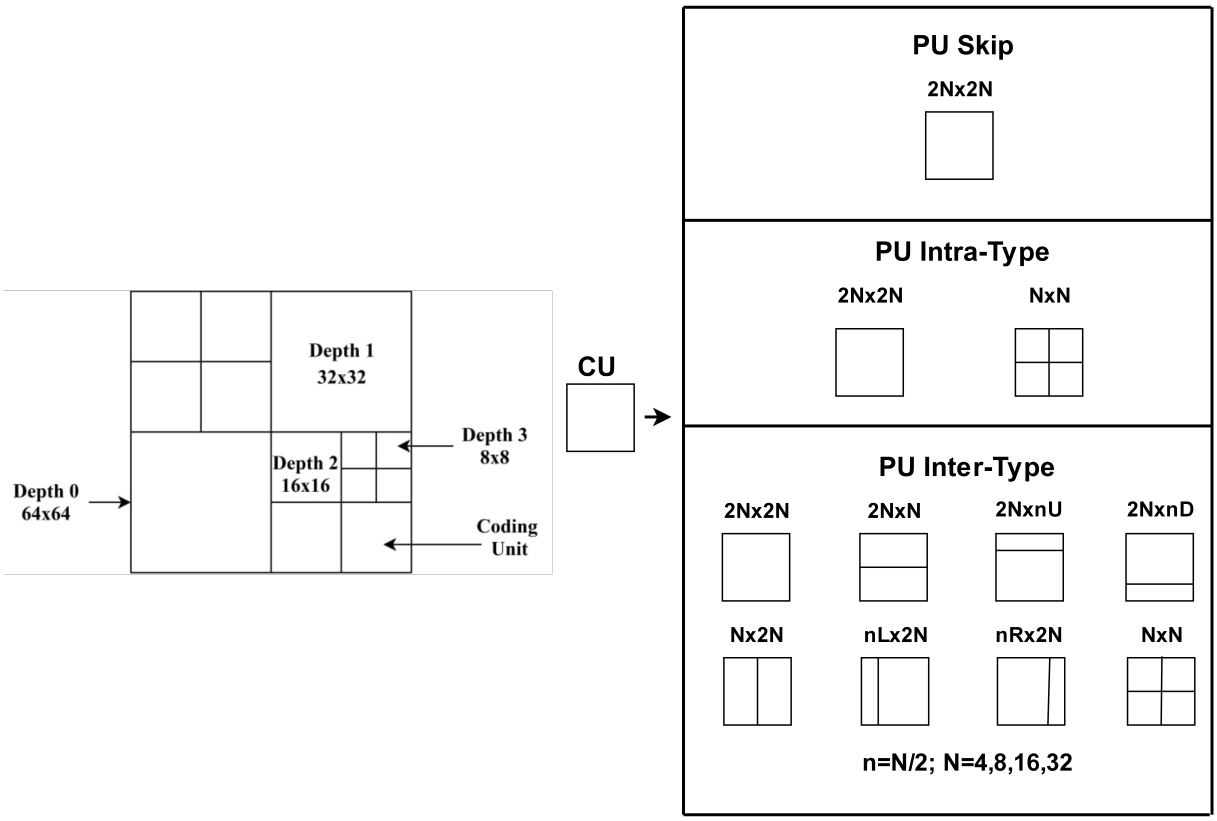


Figure 1.8 Quadtree CTU structure in HEVC

The blocks of the CTUs or CUs with a maximum size of 64x64 make up each frame. The largest block size in HEVC is 64×64, and each CTU is separated into Prediction Units (PUs) for inter-prediction, as shown in Fig. 1.8. A PU may have the following three variables: partition index (idx), partition size

(s), and partition depth (d), which are given in Table 1.2. The partition depth accepts values between 0 and 3, representing each CU size. The PU partition sizes for each CTU can be 2N×2N, N×2N, 2N×N, and N×N. The 'N' can be represented by the values 32, 16, 8, and 4, respectively. For instance, a CTU with a 32×32 partition size can include PUs with 32×32, 16×32, 32×16, and 16×16 dimensions. These are known as Symmetric partition modes. Asymmetric Partition Modes (AMPs) with PU sizes 3N/2 × 2N, 2N × 3N/2, N/2 × 2N, and 2N × N/2 are supported by HEVC in addition to these. Thus, PU sizes 24×32, 32×24, 8×32, and 32×8 can likewise exist for the CU size 32×32.

Table 1.2 List of PU partitions in 64×64 CTU

CU Size	CU depth	PU and its partition sizes
64×64	0	64×64, 64×32,64×16, 64×24, 48×64,32×64,16×64,32×32
32×32	1	32×8, 32×24, 32×16, 24×32, 16×32, 8×32, 16×16
16×16	2	16×4, 16×12, 16×8, 12×16, 8×16, 4×16, 8×8
8×8	3	4×8, 8×4

Additionally, each PU sub-partition has a partition index that starts at 0. The third column of Table 1.2 lists the different PU sizes that each CU can have. Fig. 1.9 shows this in an illustration. Also, the partition indexes for each PU sub-partition are shown in the figure. The PU modes 8×6, 6×8, 8×2, 2×8, and 4×4 are typically not used for 8×8 CU size in HEVC reference software due to the increased computational cost.

1.2.3 Transform Unit (TU)

After performing intra or inter prediction, the predicted block is subtracted from the original block to get the residual. The block transform codes the obtained residual,

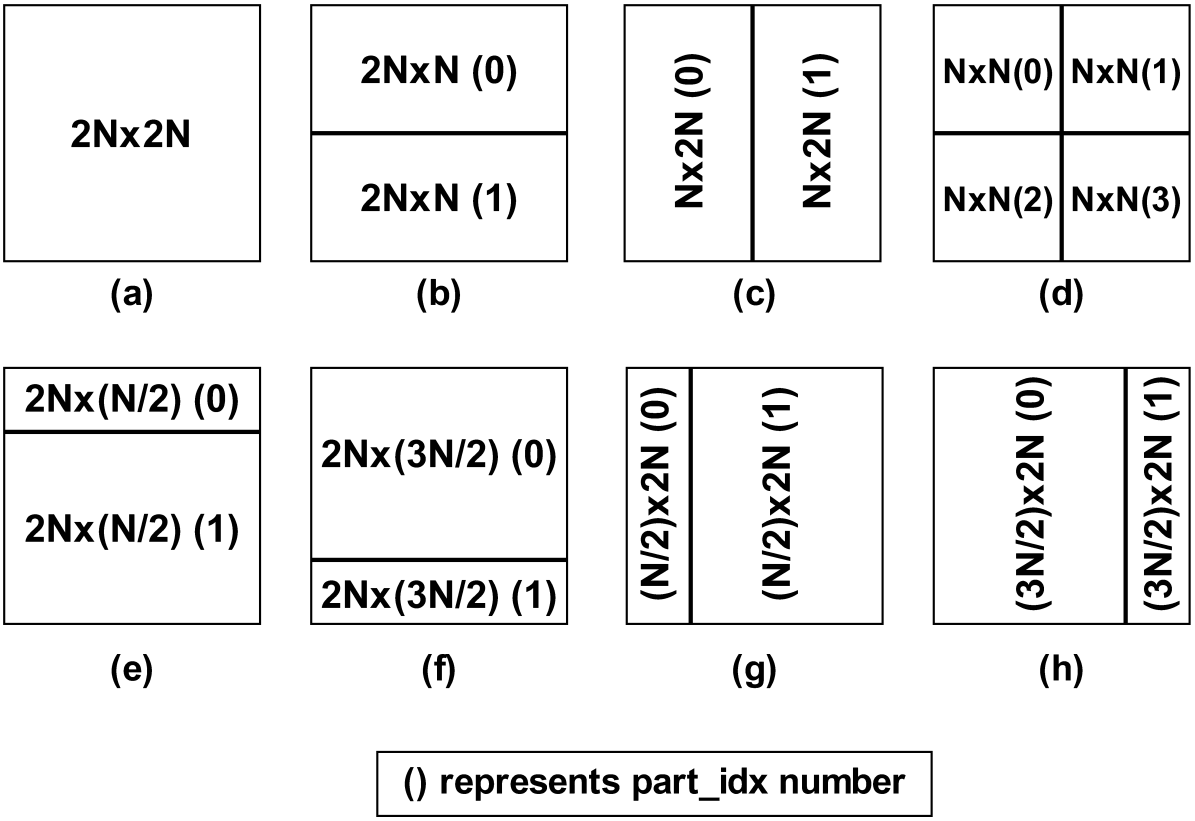


Figure 1.9 PU partition sizes with part index number for variable block size ME

and the CU serves as a root of the TU. The residual CU can be taken as TU, or it may be further split into small TUs. For the size of Transform Blocks (TBs) ranging from 32×32 to 4×4, equivalent integer basis functions to a discrete cosine transform (DCT) are defined. Moreover, the transform derived from the Discrete Sine Transform (DST) is suggested for the intra predicted residual TU with a 4×4 size.

1.2.4 Quantization

The Quantizer quantizes the resultant residual transform block coefficients. Quantization is an efficient compression tool that converts the set of values to a single quantum value. This is achieved by splitting the transformed block element-wise, and each resulting element is rounded. The quantization procedure rounds higher frequency components to zero and other frequencies to small positive or negative numbers because the human eye is more sensitive to small changes in brightness over a vast region.

1.2.5 SAO filter and Entropy coding

A nonlinear amplitude mapping is added to the inter prediction loop that seeks to reconstruct the actual signal amplitudes more accurately. The SAO filter uses the look-up table, defined via a few more characteristics found by histogram analysis at the encoder side.

Finally, the entropy coding is performed to encode the quantized coefficients, motion vector data and high level syntax elements of HEVC by using the statistical redundant information. It allows the probability modeling for more commonly utilized bits of any symbol. The Context Adaptive Binary Arithmetic Coding (CABAC) is used for the entropy coding to generate the output bit stream.

1.3 Color space representation in HEVC

The RGB color space used to capture video is not a particularly effective representation for video compression. Contrarily, HEVC uses the YCbCr color space, which divides the color space into three components and is more compatible with video coding. The component 'Y,' also known as luma, stands for brightness; Cb and Cr, also renowned as chroma, represent how much color deviates from grey towards blue and red, respectively. The luma and chroma components can be calculated by using equations (1.1), (1.2), and (1.3).

$$Y = 0.299R + 0.587G + 0.114B \tag{1.1}$$

$$C_b = 0.564(B - Y) \tag{1.2}$$

$$C_r = 0.713(R - Y) \tag{1.3}$$

The standard sampling method has a 4:2:0 structure, which means that four luma components are sampled for every chroma component because human vision is more sensitive to brightness. HEVC offers 8 or 10 bits of precision for each sample pixel value, with 8 bits being the most popular.

1.4 Motivation

The HEVC standard is developed as a successor to the AVC that uses advanced coding tools to improve efficiency. Among the advanced coding tools, the quad-tree CTU structure is key in improving efficiency. Initially, each frame is divided into the CTUs, and the structure of each CTU is determined using Rate-distortion Optimization (RDO) search process. The CTU can be considered a Coding Unit (CU) or recursively split into CUs until it reaches the smallest size. The maximum and minimum sizes of CU are 64×64 and 8×8, respectively. RDO search process consists of two stages. In the first stage, CTU is divided into 85 CUs that cover CUs of size 64x64, 32x32, 16x16, and 8x8 at depth 0, depth1, depth2, and depth 3, respectively. Then the Rate-Distortion (RD) cost is calculated for each CU and determines the CU split based on the RD cost to determine the CTU structure with optimal partitions in the second stage. This quad-tree CTU partitioning process increases the efficiency. However, the computational complexity is increased due to the testing of a large number of CU partitions.

The RD cost is calculated based on the distortion value during the RDO search process. The distortion is calculated using the Sum of Absolute Difference (SAD) metric. The SAD is obtained by calculating the absolute difference between each pixel value of the current block and reference block and then summed together. Considering the 8×8 CU block, 64 subtractions and 63 additions are required to calculate SAD. The number of SAD computations increases when the CU size increases. A large number of SAD computations increases the encoding time of HEVC. In addition, the search patterns used during motion estimation may get trapped to local minima and result in inaccurate motion vectors.

Moreover, the best partition mode must be chosen among nine inter Prediction Unit (PU) partition modes after determining the CU partitions. HEVC supports the PU partition modes like Merge/Skip mode, 2N×2N, 2N×N, N×N, N×2N, nL×2N, nR×2N, 2N×nD, and 2N×nU. The partition mode with the least RD cost is chosen as the best partition mode. The partitioning process further increases the complexity.

The above limitations are addressed in the thesis by developing the compression algorithms that integrate the traditional HEVC algorithms with the machine learning models to reduce the computational complexity in HEVC.

1.5 Problem statement

Based on the motivation, the main aim of this Doctoral thesis is to reduce the encoding time of the HEVC encoder by reducing the computational complexity of the RDO search process and motion estimation.

1.6 Research objectives

The following goals are defined to achieve the main objective.

1. To develop an efficient algorithm that overcomes the local minima and accelerates the motion estimation process.
2. To develop a method that reduces the complexity in determining the CTU structure using a machine learning approach in HEVC and SHVC.
3. To develop a method for fast encoding of Region of Interest (ROI) in surgical video sequence with high quality using SHVC.

1.7 Thesis Organization

The rest of the thesis is structured as follows:

Chapter 2 presents the background work on motion estimation and CU size prediction in HEVC. It also presents different methods proposed by the authors to determine the CTU structure in Scalable HEVC.

Chapter 3: Presents the Multi-Level Resolution Vertical Subsampling (MLRVS) algorithm to reduce the computational complexity of the motion estimation. Also, the complexity reduction method is developed to reduce the encoding time of HEVC.

Chapter 4: Presents the fast Convolutional Neural Network model to predict the CTU structure. Besides, the MLRVS algorithm proposed in the previous chapter is used along with the Cross Diamond Octagonal search pattern to further reduce the encoding time.

Chapter 5: Presents the CNN +LSTM deep learning approach in Scalable HEVC for fast Coding Unit size decision. In addition, Horizontal Subsampling Motion Estimation

(HSME) is used to accelerate the motion estimation process.

Chapter 6: Presents the KCF Tracker-based Region of Interest encoding using Scalable HEVC for Surgical Telementoring applications. Simulations followed by a comparison of results with existing works are also presented.

Chapter 7: Presents the region-based motion estimation using YOLOv4 and next frame prediction using a deep neural network.

Chapter 8: Draws conclusions from the earlier chapters and concludes the thesis.

Chapter 2

Review of Literature

2.1 Introduction

The increased demand for compression of high-definition videos with high efficiency leads to the development of the High Efficiency Video Coding (HEVC) standard. HEVC uses effective compression techniques to produce a twofold gain in video efficiency over H.264/AVC. One of the most computationally demanding blocks in video CODEC is motion estimation (ME). Due to a big processing unit CTU and dynamic segmentation of the prediction unit in HEVC, the complexity of ME further rises. Moreover, determining the CTU structure in HEVC increases the encoding time. The computational complexity in deciding the CTU structure is increased due to the conventional RDO search process. The complexity further increases in SHVC due to the multiple layers, such as Base Layer and Enhancement Layers.

Several authors proposed various methods to reduce the computational complexity in HEVC and SHVC. This chapter discusses the related work on ME, CU size prediction in HEVC, and SHVC in the following sub-sections.

2.1.1 Related work on Motion Estimation

Hsieh et al. [5] developed a power-efficient ME controller to reduce power dissipation. The dissipation is due to the large coding bandwidth required to access the current or reference pixel values during the ME process. Vayalil et al. [6] proposed a method that

uses the snake scan to get the data of a row or column. In addition, the residue number system is used to improve the speed of the Sum of Absolute Difference (SAD) calculations. This approach helps to decrease the encoding time. Cebrián- Márquez et al. [7] use the pre-analysis stage, which performs block-based ME to estimate Rate-Distortion cost. The estimated cost is used to build the optimal quadtree by omitting many unnecessary partitions, reducing the encoding time. Fan et al. [8] take the conventional HEVC merge mode's motion vector as a center and apply the ME process around it and along the axis in a small search region. This approach improves bitrate saving. However, the encoding time is increased.

The Test Zone (TZ) search algorithm in HEVC uses multiple search points at the start, making it difficult for real-time implementation. Pakdaman et al. [9] use a single search point at starting of the TZ search algorithm. The wavelet transform is used to analyze the current and reference frames. After analyzing, similar points are identified and matched to determine the single search point. Jiang et al. [10] proposed the approach to predict the optimized motion vectors by utilizing the motion consistency of the adjoining PUs. Similarly, the spatial correlation of neighboring CUs can be utilized to forecast the depth of the current CU. Gogoi and Peesapati et al. [11] proposed a hardware architecture for ME using a hybrid search pattern. The hybrid search pattern consists of hexagonal and square global patterns and two, three, and four-point local search patterns. This method minimizes the encoding time by 11%.

In [12], a GPU-based minimal delayed parallel ME approach is developed. The authors considered the quadtree coding structure of HEVC to facilitate the parallelization hierarchically by optimizing the ME process in the CTU, PU, and MV layers. A unique motion vector predictor decision scheme is specifically suggested for the CTU layer to mitigate the negative consequences of erroneous MV prediction by eliminating the CTU-level dependencies. An innovative indexing table is created for the PU layer to implement an effective cost derivation technique. As a result, it is possible to compute every PU's cost easily and effectively. In [13], an effective ME design with a coordinated algorithm and architectural optimization is proposed. A predictive integer ME (IME) algorithm is suggested to choose the most likely search direction and lower the number of search points by 90.5% to reduce complexity. Additionally, a Fractional ME (FME) method based on PU size is used to lessen the interpolation filtering. Moreover, interlaced scheduling is

used to cascade the IME and FME computations. The authors in [14] developed a low-power ME algorithm and HEVC architecture for consumer applications. The suggested algorithm and architecture use sub-sampling, pixel truncation, data reuse, and heuristic search range approaches to reduce processing resources.

In [15], for texture and depth 3D video, a motion-information-based three-dimensional (3D) video coding methodology is proposed. The camera motion and depth information is used to project the surrounding temporal texture and depth frames into the location of the current frame, helping the encoder enhance its RD performance. The projected frames are then included in the reference buffer list as virtual reference frames. The number of bits needed to represent the residual will be drastically reduced since these virtual reference frames might resemble the current frame that needs to be encoded more than the traditional reference frames.

Park et al. [16] suggested a quick encoding technique to effectively reduce the encoding overhead of Affine ME (AME) when using a Multi-Type Tree (MTT). The suggested approach follows a two-step process. The first step uses an early termination method that takes advantage of parent CUs to eliminate unnecessary AME and Affine Motion Compensation (AMC) procedures. It specifically uses motion data from the parent CU that is already encoded. Then the number of reference frames used for AME is decreased in the second step. Trudeau et al. [17] combine the ME with a Successive Elimination Algorithm (SEA) that minimizes the number of computed cost functions without affecting rate or distortion in the context of ME for video coding. The SEA using the SAD metric eliminates the MV candidates in the search region that can't overcome the current minimum. Moreover, the authors suggested a dynamic method that generates cost-based search orderings, eliminating 61% of the block-matching loop iterations carried out by the Rate-Constrained SEA method (RCSEA).

In [18], a ME technique to simplify the H.265/HEVC encoding process is provided. First, all PUs are divided into two classes, namely parent PUs, and children PUs, following the type of PU partitioning. The proposed approach uses the best MV selection correlation between child CUs and parent CU to perform the Block Matching (BM) search operation. The BM search process of the children's PUs is then adaptively skipped if their parent PU selects the beginning search point as its final optimal motion vector in the ME process. [19] introduces an innovative Zoom ME and motion compensation ap-

proach based on the local area scaling method. A zoom vector (ZV) is proposed that can either grow or shrink a reference block size to express and encode the zoom motion. Block matching is then carried out by interpolating the resized reference block to the size of the present block. In addition, a 3-D diamond search pattern is introduced to reduce the search points and accelerate the Zoom ME.

He et al. [20] described a VLSI implementation of Fractional ME architecture in HEVC for applications requiring UHD video. A bilinear quarter-pixel approximation is suggested to simplify the fractional search processes. Additionally, a data reuse method is used to lower the hardware expense of the transform. Moreover, the algorithm is fully pipelined and achieves greater hardware utilization by employing pixel parallelism and a separate access pattern to memory. The researchers in [21] suggest a new Integer ME method that drastically narrows the search space. The computational cost of the proposed technique is decreased without noticeably degrading coding efficiency by acquiring additional precise Motion Vector Predictors (MVPs) in a bottom-up order and scanning confined regions around multiple MVPs. These MVPs are derived from PUs in the CUs at the hierarchically lower level.

In [22], an ideal fast ME approach based on a Rate-Complexity-Distortion (R-C-D) analytical model is proposed. The ME complexity is added to the conventional R-D model that clearly describes the R-D performance under various complexity budgets. The proposed approach determines the R-C-D optimum search ranges for a few representative PUs based on the R-C-D model, which is then dynamically expanded or refined in accordance with motion characteristics. The suggested method achieves R-D performance nearly equal to a full search. Hu and Yang [23] reduce the complexity of HEVC by initially constructing the ME as a statistical inference problem at the integer pixel level and then suggesting the confidence interval-based ME (CIME) approach. The proposed CIME method can be used in lieu of the current fast search implemented in HEVC to reduce the complexity.

These algorithms use fast search patterns to reduce the number of search points. However, there is a possibility of converging to local minima due to the early termination of the searching process. Moreover, the encoding time decreased by the state-of-the-art motion estimation approaches is small. Hence there is a scope to further decrease the encoding time of the encoder.

2.1.2 Related work on CU size decision in HEVC

Shen et al. [24] proposed a method that gathers information from four spatially nearby CTUs and a co-located CTU to narrow the CU depth search area. CTUs in the vertical and horizontal directions are considered more important and are utilized for depth prediction. The current CU is classified into one of five types based on the prediction result, ranging from homogenous to fast motion zones. The depth is restricted based on the classification outcome. Correa et al. [25] divide the frames into constrained frames (Fcons) and Unconstrained frames (Fun-cons). The standard RDO search process in HEVC encodes the unconstrained frames. In contrast, the rest of the frames are encoded using the largest CTU depth of the co-located CTU in the preceding frame to reduce the CTU partitioning complexity. The number of the constrained frames (Fcons) between the Fun-cons is determined based on the complexity. Correa et al. [26] use the data of spatially surrounding CTUs and the co-located CTU in the last constrained frame to define the maximum depth of the current CTU.

Bae and Sunwoo [27] method save CU depth statistics from CTUs in the preceding five to six frames and utilize it to determine when to terminate the CU and PU. A weighted framework is used to give greater weightage to the nearby frames. If all the depths are equal, the current CU and PU search is performed with the very same depth as the final depth. Otherwise, CTU statistical properties are estimated and employed in the decision-making process.

Shen et al. [28] use the four spatially adjacent CTU information to classify the CTU into different classes spanning from homogeneous to the rich texture. The texture of CTU is determined based on equation (2.1).

$$D_{prev} = \sum_{j=0}^{3} \alpha_j D_j \tag{2.1}$$

Where, D_j=Depth value, α_j= weighting factor of each adjacent CTU
CTU belongs to type 1 if $D_{prev} < 0.5$, type2 if $0.5 < D_{prev} \leq 1.5$, type3 if $1.5 < D_{prev} \leq 2.5$ and type4 if $D_{prev} > 2.5$. The maximum CTU depth is also decided based on D_{prev}.

The authors in [29] determine the CU depth by using the average depth information of the spatially co-located CTUs. The obtained information is compared to the threshold value to determine whether the depth range belongs to {0,2} or {1,3}. In [30], the authors

limit the search range of the present CU between maximum and minimum depths obtained from the top and left CTUs. The present CU split decision is taken based on the depth value. Moreover, the early termination algorithm based on the RD cost is developed for all intra configuration that exploits the temporal correlation between neighboring frames. The researchers in [31] decide on the CU's splitting by performing a two-stage prediction. In the first stage, the two most common CU depths from the previously encoded frame are derived to form the CUControlSet. Then the depth level in each CU at the second stage is obtained from the Co-located CU in the previous frame to form the CUPredictSet.

$$CUAllowableSet = CUControlSet \cup CUPredictSet \tag{2.2}$$

The CU split, if the current depth is present in the CUAllowableSet and then performs RD computations for each partition. In [32], the authors decide the depth level of the CTU by using the spatially neighboring CTUs and the co-located CTU. The segmentation process of CTU terminates at depth 0 if

$$\sum_{n=0}^{4} C_n \lambda_n \geq \alpha \tag{2.3}$$

Where λ_n=Weighting factor, C_n=weight of reference CTUs.

In [33], Neighboring Mean Squared Error (NMSE), Direction Complexity (Dcom), Sub-CU's complexity Difference (SCCD), and Quantization Parameter Step (QStep) features are used to determine the complexity of CU. Initially, the CUs are classified into groups, namely Group A, Group B, and Group C, with high, small, and medium texture complexities. For each group, different strategies are used to split the CU, such as Skipped strategy for Group A and early termination for Group B and Group C with the actual encoding process. Based on the features derived, the SVM is used to classify the CUs. The authors in [34] proposed the algorithm that selects the CU using offline trained SVM and online trained Bayesian approach. The features such as context, texture, and CU data like QP (Quantization Parameter) and 2N×2N mode information are extracted to train the SVM and Bayesian models. The offline and online trained models, along with the skip mode detection techniques, are used to accelerate the CU selection process.

Mallikarachchi et al. [35] use both motion and threshold-based CU partition approaches. During motion-based CU classification, inter RD cost of N×N, CU size, and motion complexity information is used. In the threshold approach, RD cost of inter N×N,

Quantization parameter, and CU size data are used to decide whether to split the CU. In addition, the motor vectors are reused to accelerate the ME. This approach efficiently reduces the encoding time.

Kuo et al. [36] used the Predictive Coding Unit Depth (PCUD) to determine the depth range. The PCUD is the maximum depth of the left, top left, top, and co-located CTU of the previous frame. Then the PCUD is compared to the threshold values to detect the depth range such as [0,1],[1,2],[0,2], and [1,3]. Moreover, the Texture-Aware Splitting Termination is used for early CU depth decisions by analyzing the correlation between the block's texture and the CU size. In [37], the authors developed the two-level SVM model to determine the CU depth. At the first level, the features like texture, RD-cost, and context data are used to train the SVM at different CU depth levels. The features are generated by using the HEVC reference software. The trained SVM is used to find the CU depth. If SVM at level 1 fails, the SVM at level2 is used to predict the CU depth. The SVM at level2 is trained using the context, texture, and coding information of CUs.

Shen et al. [38] reduce the encoding time by considering the correlation between the frames. Firstly, the prediction mode information between the depth levels is explored, and then the coding data between the neighboring CUs in the same frame is analyzed. Finally, the coding information of co-located CUs in the current and previous frames is derived. All this information is used to reduce the complexity of the encoder. In [39], the CU size is determined based on texture homogeneity. For CUs with homogeneous texture, early CU size decision and skip intra search is performed on CUs with small size. The CU depth level for complex CUs is determined by collecting coding information from neighboring CUs and then combine with texture property. In [40], the partitions in the CTU are decided based on the edge complexity information in various directions. The local and global edge complexity data determines the CTU structure.

In [41], the decision on CU split is taken by calculating the correlation between the motion divergence and the CU splitting using the motion vectors of pixels. This process requires extra hardware resources due to the calculation of pixel-wise optical flows, increasing the encoder's overall complexity. Bouaafia et al. [42] use the Support Vector Machine (SVM), and Convolutional Neural Network (CNN) approaches to predict the CU partitions during the Rate-Distortion Optimisation (RDO) search process. This approach reduces the encoding time.

Huang et al. [43] proposed the RD complexity optimization scheme to preselect the CU depth and speed up the Transform Unit (TU) tree decision process. Moreover, the early Prediction Unit (PU) and CU termination algorithms are provided to decrease the encoding time. Lu et al. [44] minimize the complexity of the encoder by generating the classification trees. The trees are generated using the intra and inter features obtained after encoding the conventional HEVC algorithm. The features provide the context and texture properties of PU, CU, and TU. Sharma and Arya [45] optimize the parameters of the HEVC using the non-dominated sorting genetic algorithm II to improve the compressed video quality. This approach concentrates on increasing the quality of video and decreasing the file size. Yan et al. [46] reduce the complexity of the intra-prediction by using the statistics of the rough mode decision method. In this process, the number of most probable modes is decreased based on the PU size. The decrease in the most probable modes decreases the complexity of the encoder.

Moreno et al. [47] have presented an algorithm that minimizes the encoder complexity by deciding the CU size based on the early termination condition. Kim et al. [48] have proposed a method that bypasses the interpolation process of list 1 when the bi-predicted motion data of list 0 and list 1 are the same. This strategy lessens the intricacy of encoder and decoder. Lee et al. [49] have described an early skip mode scheme to reduce the encoder's coding time. Ahn et al. [50] use the spatial and temporal parameters to reduce the encoder's coding time. Here, the decision of subdividing the CU is taken based on the motion and texture complexity. Purnachand et al. [51] have developed an algorithm that only omits the global search step when the cost difference between the Initial Search Point and the current bock is lower than the threshold. The threshold value, in this case, is the lowest cost of temporal and spatially co-located blocks. By using this method and rotating hexagonal search pattern, the complexity of the encoder is decreased.

2.1.3 Related work on CU size prediction in SHVC

The methods proposed by different authors to minimize the complexity of SHVC are discussed below.

In [52], the authors derive the depth probabilities using the Bayesian approach by combining the CU's data and the correlation degree between the frames. This informa-

tion is integrated with the texture-based all-nonzero and all-zero blocks using Lagrange Interpolation Polynomial to terminate the depth prediction process early. Also, the mode probabilities are derived and mixed with the Jarque-Bera test to skip the intra-mode prediction. In [53], the researchers determine the partitions of the CTU early by using the Decision Tree classifier. The features such as global texture, local texture, and context information are used in the proposed technique to determine the partition structure. Moreover, the one-dimensional gradient descent search is used for fast intra-mode decisions. In [54], the authors use Adaptive Mode Pruning (AMP) and Mode-dependent termination for fast intra-mode decisions. Initially, the priority of the Intra Block Copy (IBC) and Intra Sub-partitions coding (ISP) modes are checked, and if the IBC priority is high, the linear modes prediction is not performed. Otherwise, the ISP mode is dropped adaptively based on the texture and correlation strength. Then the CU is categorized and avoids the prediction of unnecessary depths using the early termination model.

Liu et al. [55] used a K-means method to determine the CU depth level by classifying the thirteen neighboring CTUs into three cases. Tohidypour et al. [56] increase the performance of SHVC with quality scalability; however, it restricts the number of EL layers to one. Chen and Chang [57] use the Rough Mode Cost (RMC) to determine the depth of CU and TU partition. The PU modes with larger RMC values are excluded from the candidate list, which results in a decrease in coding time. The authors in [58] developed a method that applies Selective Encryption over the uniformly distributed bitstream elements to preserve the statistical properties and the length of the SHVC encoded output. The researchers in [59] use the spatial and temporal correlations and the correlation degree to determine the candidate depths. Then, they use the Rate-Distortion costs and the residual coefficients to terminate the depth and mode selection process early. Tohidypour et al. [60] use the combination of machine learning and the Bayesian approach to decrease the SHVC complexity. Shen and Feng [61] decrease the complexity of SHVC by utilizing the information of motion activity and mode complexity to skip the unnecessary CU size and mode search.

Hsuan et al. [62] reduce the coding time of EL by using the encoded information of prediction modes, CU sizes, Rate-Distortion (RD) Costs, and motion vectors of BL and CU sizes of the EL. Authors in [63] decrease the complexity of SHVC by using the collocated BL and the four neighboring CUs information to determine the current CTU's

block depth in EL. In [64], the CTU encoded in BL utilizes the structure of the collocated CTU present in the previous frame or the neighboring four CTUs present in the same frame.

Wang et al. [65] suggested a strategy for Quality SHVC, which excludes depths with low probability. Additionally, they provide an early depth decision strategy and only allow checking a portion of the 35 defined Intra modes rather than all. Tohidypour et al. [66] predict the current CTU structure in EL using the already decoded CTU information in BL and EL. They build a model using the Bayes rule and then train it with a small dataset, improving the coding speed. The method developed in [67] adaptively chooses the best motion search range for the EL based on the correlation between BL and the EL. Moreover, they avoid redundant computations and also terminate the mode search in the optimal search range early to reduce the coding complexity. In [68], the authors developed the complexity reduction method for SHVC based on the RD_cost information of the neighboring CU in the EL and the RD_cost of co-located CU in BL. The optimal threshold is selected to determine the CU depth based on the RD_cost and Quantization Parameter (QP) value. In [69], the researchers explored the relationship between the present CU and the neighboring CUs to find the less frequently used modes. These modes are skipped to reduce the complexity of SHVC.

The approaches discussed in section 2.1.2 and 2.1.3 concentrates on decreasing the computational complexity of the RDO search process in HEVC and SHVC. Some of the approaches used a very small training dataset, which led to the prediction of the CTU partitions inaccurately. In addition, most algorithms exploit only spatial features of the image. Moreover, the motion estimation method used in the encoder increases the encoding time. Hence an efficient model is required that decreases the complexity of the motion estimation and RDO search process.

Chapter 3

Multi-Level Resolution Vertical Subsampling and Early Skip Mode Detection Algorithm

3.1 Introduction

The High Efficiency Video Coding (HEVC) or H.265 [70] standard compresses ultra high definition video sequences with approximately 50% less bitrate than the H.264/Advanced Video Coding standard while maintaining the same video quality [71]. The video compression involves splitting the frame into Coding Tree Units (CTUs), intra and inter predictions, finding transform and quantization for the residual block, and filtering operations using deblocking [72] and Sample Adaptive Offset (SAO) [73] filters. HEVC is widely used in ultra high definition online video streaming and surveillance applications, which are shown in Fig. 3.1. In HEVC, complexity increases along with an increase in efficiency and most of the time is consumed during the process of finding the RD cost for Prediction Units (PU), and the motion estimation process.

Rui Fan et al. [74] suggested a technique that utilizes the Priority Guided Fast Partial Internal Early Termination algorithm and motion complexity. The PU is categorized here based on motion, i.e., smooth, medium, or complex motion. Pan et al. [75] have presented a new algorithm called adaptive Fractional Pixel Motion Estimation skipped algorithm. Here the children type PUs can be encoded based on the best motion vector of root PU using Integer Pixel Motion Estimation. These algorithms use one of the search pattern such as Three Step Search (TSS) [76], improvements in TSS [77–79], logarithmic search

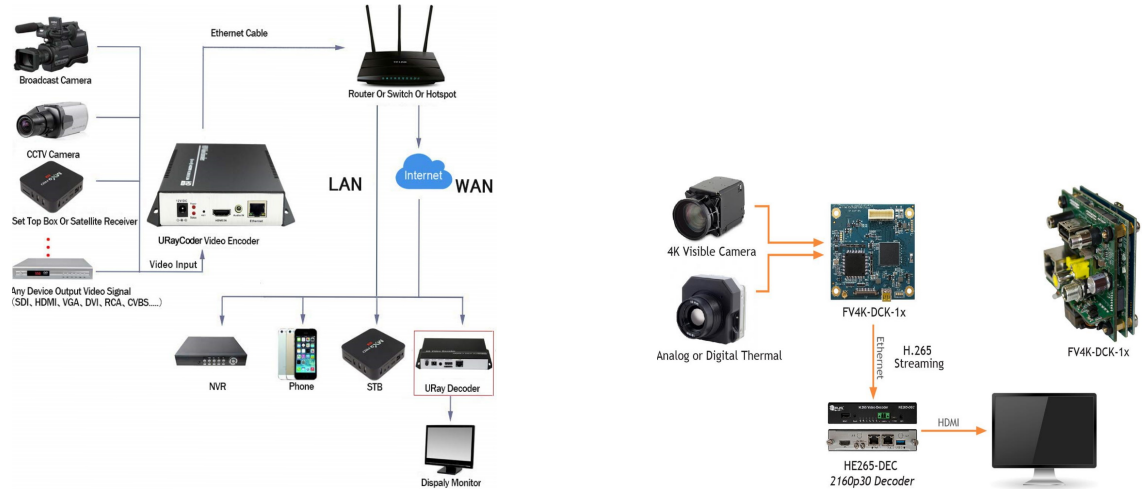


Figure 3.1 Applications of HEVC: (a) Online video streaming (b) Surveillance

[80], One dimension full search [81], etc., to speedup the motion estimation process. These algorithms may reduce complexity by reducing the number of search points. However, there is a possibility of converging to local minima due to the early termination of the searching process. We have developed a Multi-Level Resolution Vertical Subsampling (MLRVS) algorithm to prevent the early termination of the searching process and improve the motion estimation speed.

The MLRVS algorithm using vertical subsampling, and the complexity reduction algorithm helps to reduce the encoding time of the encoder. In addition, New Cross Diamond Diamond (NCDD) and New Cross Diamond Hexagonal (NCDH) search patterns are proposed to accelerate the motion estimation process.

3.2 Overview of motion estimation process during inter prediction in HEVC

In HEVC, each frame is segmented into CTUs. It is possible to subdivide each CTU [82] into coding units or the CTU itself as the CU. The size of CU can be 64, 32, 16, or 8. The CU contains one luma Coding Block (CB) and two associated chroma CBs. Every CU can be additionally partitioned into Prediction Units (PU). The size of PU should be less than or equal to the size of CU. The structure of CTU is shown in Fig. 3.2.

The CTU can be split up to a maximum depth of four. HEVC supports the PU partition modes like Merge/Skip mode, 2N×2N, 2N×N, N×N, N×2N, nL×2N, nR×2N, 2N×nD, and 2N×nU. The RD cost can be determined for PU partition modes by utilizing the equation (3.1).

$$J = D + \lambda \times R \tag{3.1}$$

Where, R→ Number of bits required to transmit, $\lambda \rightarrow$ Lagrangian multiplier, D =Distortion.

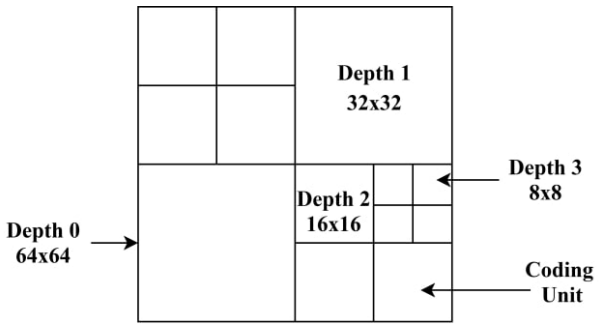


Figure 3.2 Structure of CTU with coding unit depths ranging from 0 to 3

The distortion or SAD is calculated during the motion estimation process to find the RD cost. Motion estimation in HEVC plays a vital role in decreasing the bit rate for storing or transmitting the video signal. The TZ search algorithm (discussed in section 3.2.1) is used for motion estimation in HEVC. During the motion estimation process, for every block in the current frame, the appropriate matching block can be found in the previous frame inside the search area. Generally, SAD is the widespread matching criterion to find the distortion, which is used to find the best matching block in the previous frame for the block in the current frame. SAD is obtained by first calculating the absolute difference between each current block pixel and the corresponding pixel in the reference block. Then these differences are summed together to get the final SAD value. The SAD is calculated by using equation (3.2). Then the RD cost is calculated for the partition modes using the SAD value. Among PU modes, the best mode is the mode that has a lower RD cost.

$$SAD = \sum_{k,l} |S_A(k,l) - S_B(k,l)| \tag{3.2}$$

Where, $S_A(k,l)\rightarrow$ (k, l)th pixel in current frame-block, $S_B(k,l) \rightarrow$ (k, l)th pixel in reference frame-block. Both current frame-block and reference frame-block are equal in size.

3.2.1 TZ search algorithm

The TZ Search algorithm [83] is explained in the following steps

1. First, calculate the median predictor (discussed in section 3.2.2).
2. After calculating the median predictor [84], check whether the zero motion vector is the best starting point than the median predictor.
3. Now consider the median predictor as an initial starting point and perform the first search.
4. In the first search, either diamond search [85] or square search patterns can find the best motion vector. Here the search window can have a minimum distance of one to maximum distance of search range. The distance at which the point with minimum distortion occurs is considered 'Best Distance (BD).'
5. Now take the Best distance and check the following three conditions.
   - If the BD is zero, the searching process stops
   - If 1 < BD < iRaster, perform refinement directly.
   - If BD> iRaster, perform a raster scan by taking the value of iRaster as a stride length.

   The raster search process can be done on a whole search window if the difference between the starting position and the first phase motion vector is too significant.
6. If the best distance in the previous search is not zero, apply the star or raster refinement. During this refinement stage, the last search's best motion vector is taken as the starting point. Here the distance is in the range of one to search range. Diamond or square search patterns are used in the refinement process. During star refinement, the distance is multiplied by two in each iteration until it reaches the

search range. During raster refinement, the distance can be divided by two in each iteration until it goes one.

3.2.2 Median calculation in HEVC

In HEVC, the median predictor is obtained using the Predictors A, B, and C. Predictors A, B, and C are the left, top, and top right predictors for the median predictor, as shown in Fig. 3.3. The median predictor is calculated by using equation (3.3).

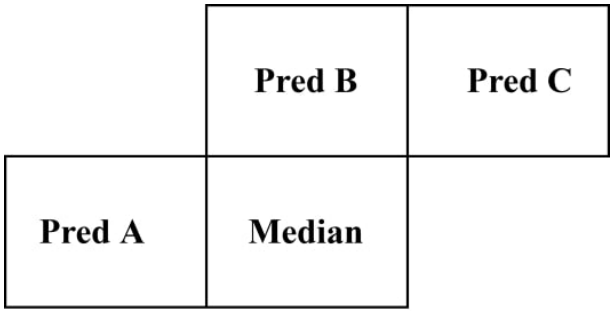


Figure 3.3 Median predictor prediction using left, top, and top right predictors.

$$Median(A, B, C) = A + B + C - Min(A, Min(B, C)) - Max(A, Max(B, C)) \quad (3.3)$$

3.3 Proposed Work

The framework of the proposed method is shown in Fig. 3.4. The encoding process involves motion estimation, inter prediction, transform, quantization, and entropy coding to generate the bitstream. We proposed the MLRVS algorithm to accelerate the motion estimation process. The algorithm involves vertical subsampling, and motion estimation using newly proposed search patterns in the vertical sub-sampled frames. Besides, a complexity reduction algorithm is used during the inter-prediction and quantization process to reduce the encoding time. The MLRVS algorithm and the complexity reduction algorithm are explained below.

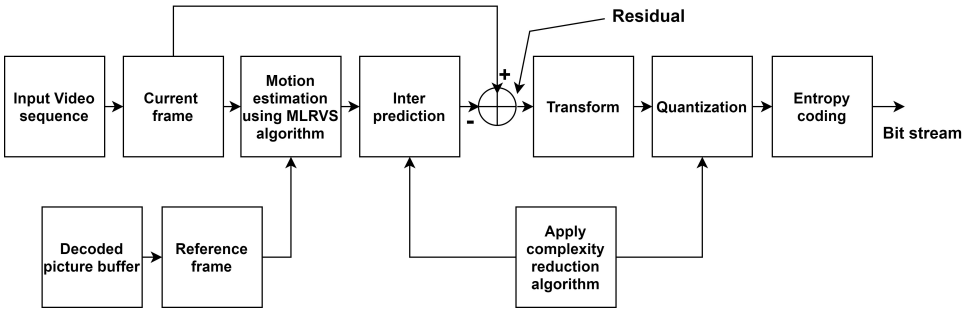


Figure 3.4 The framework of the proposed method.

3.3.1 MLRVS algorithm

The MLRVS algorithm shown in Algorithm 3.1 is explained in the below steps.

Algorithm 3.1: MLRVS algorithm

Input: Prediction Block (P), Search range (Sr), Motion Vectors=(MV, MV1, MV2), zero MV and Neighbours= (MV(0,0) and MVx, MVy, MVz),Frame (Orig), Best Distance (BD), Distance=(D1, D2, D3)

Output: Best Motion Vector (BMV)

Initialization : MV=(0,0); TotalCost=∞

1: **for** $tmpMV$ **in** (MV, MVx, MVy, MVz) **do**
2: $tmpCost = getCost(tmpMV, Sr, P)$;
3: **if** $tmpCost < TotalCost$ **then**
4: $TotalCost \leftarrow tmpCost$;
5: $MV \leftarrow tmpMV$;
6: **end if**
7: **end for**
8: $DB = \{1, 2, 4\}$;
9: $HalfResolutionframe(HR) = evenRows(Orig)$;
10: $QuarterResolutionframe(QR) = evenRows(HR)$;
11: $(D1, MV1) = SearchPattern(BD, MV, Sr, P, QR)$;
12: $(D2, MV2) = SearchPattern(D1, MV1, Sr/2, P, HR)$;
13: $(D3, BMV) = SearchPattern(D2, MV2, Sr/4, P, Orig)$;
14: $\rightarrow$ ($Searchpattern$ can be $NCDD$ or $NCDH$)

15: $BD = D3$**;**

16: if BD = 0 then

17: *Stop the searching process*

18: else

19: *Perform refinement operation*

20: end if

Step1 Find the median predictor using equation (3.3).

Step2 After median prediction, extract the Half Resolution (HR) and Quarter Resolution (QR) frames using the frame extraction. To create the HR and QR frames, vertical subsampling is used. The representation of the vertical subsampling frame extraction process can be observed in Fig. 3.5. Initially, the original frame (Orig) of (M×N) size is taken and subsampled. M and N represent the number of rows and columns of the frame. The vertical subsampling is used to reduce the resolution of the original frame. The HR frame, which is of size (M/2 × N), is obtained by considering the original frame's even rows, and the QR frame is obtained by considering the even rows of the HR frame.

Step3 After extracting QR and HR frames, apply the motion estimation process to find the Best Motion Vector (BMV). In this algorithm, the BMV is obtained by first calculating the Motion Vector MV1 in the QR frame. We use the search patterns like NCDD or NCDH to find the motion vector. After the searching process, take the MV1 of the QR frame as an initial search point in the HR frame and find the Motion Vector MV2 in the HR frame. Finally, take the MV2 of the HR frame as an initial search point in the original frame and find the original frame's Best Motion Vector (BMV) using the search pattern.

Figure 3.5 Frame extraction process

The advantage of this process is the possibility of converging towards local minima is significantly less.

3.3.2 Complexity reduction algorithm

This section presents the complexity reduction algorithm to decrease the H.265 encoding time. The flowchart representing the complexity reduction algorithm is shown in Fig. 3.6.

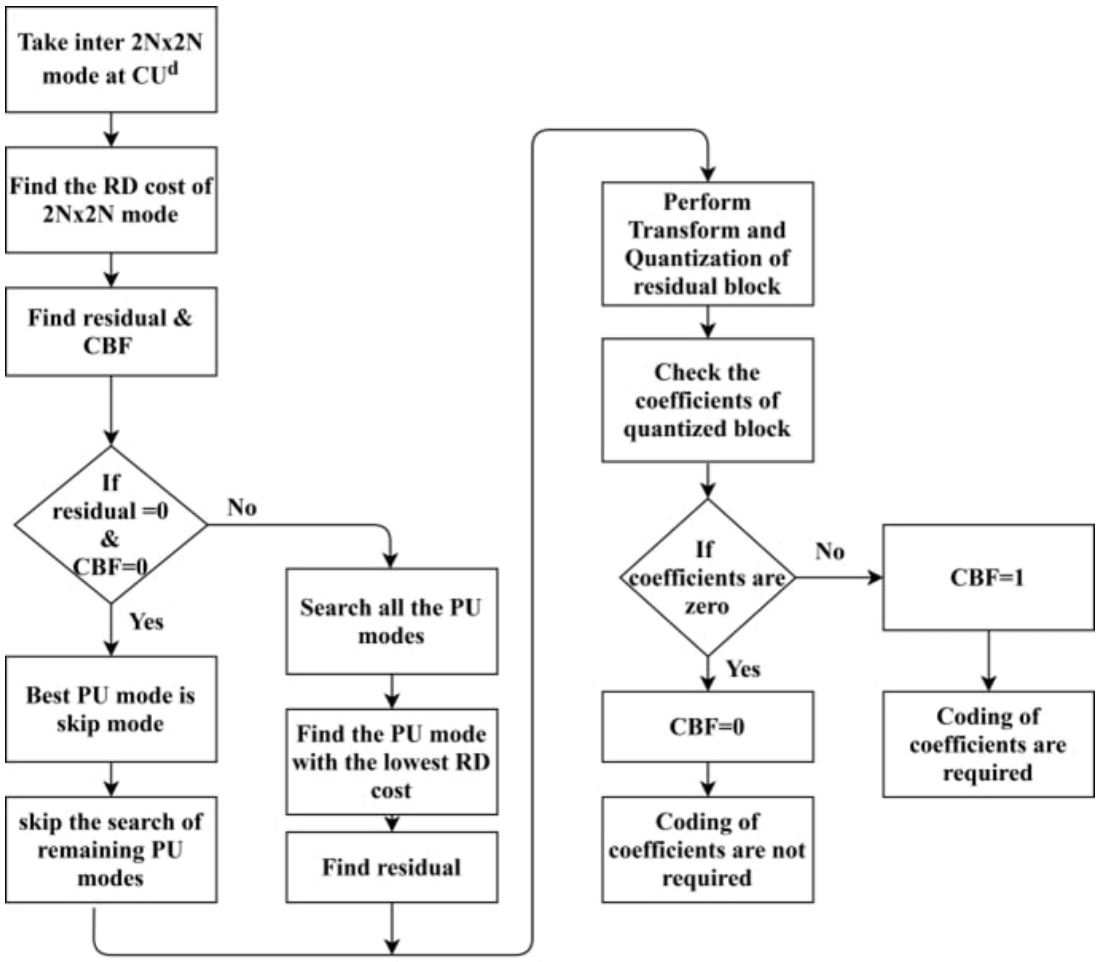


Figure 3.6 Flowchart of the complexity reduction method.

The CUs of size 2N×2N at each depth are taken, where N can be 4, 8, 16, or 32. Then the RD cost (J) is calculated by using equation (3.1). Let 'J_C' represents the RD cost of Current CU and 'J_B' represents the RD cost of Best CU. Here Best CU is the CU having lower RD cost. The Best CU cost is determined based on the condition in (3.4).

$$J_B = \begin{cases} J_C, & if\ J_C < J_B \\ J_B, & otherwise \end{cases} \tag{3.4}$$

Now update the prediction data and reconstruction data.

Conventionally to determine the skip mode, check whether the RD cost of the skip (J_s) is less than the RD cost of merge mode (J_m) or not. Let 'N' represents the maximum number of merge candidates and skip candidates. The number of merge candidates is signaled in the slice header. Usually, the 'N' value is five. The merge RD cost at each depth can be calculated using equation (3.5).

$$J_{m_d} = \frac{1}{N} \sum_{k=0}^{N-1} J_{m_{u-k}} \tag{3.5}$$

Similarly, the skip RD cost J_{s_d} can be calculated using equation (3.6).

$$J_{s_d} = \frac{1}{N} \sum_{p=0}^{N-1} J_{s_{v-p}} \tag{3.6}$$

Here, 'd' is the current CU depth, u and v represents the number of merge modes and skip modes treated as best PU modes for particular CU size. $J_{m_{u-k}}$, $J_{s_{v-p}}$ represents the rate-distortion cost of k^{th} merge mode and p^{th} skip mode.

During block merging, the Merge flag specifies that block merging is utilised to get the motion data for PU. Merge index is used for determining the candidate present in the merge list. In block merging, the skip mode with the skip flag is incorporated.

If $J_s < J_m$, then skip the computation of RD cost for the remaining modes. Otherwise, perform the RD computation for all other PU modes.

To determine the skip mode early, after performing the RD computations, the early skip condition is checked by gathering the Coded Block Flag (CBF) and residual information. The skip condition is shown in equation (3.7).

$$Skip\,mode = \begin{cases} True, & if(residual = 0\,and\,cbf = 0) \\ false, & otherwise \end{cases} \tag{3.7}$$

Here CBF is used to indicate whether the Transform Block (TB) has any significant non-zero coefficients or not. Generally, after calculating the prediction residual, each CU is divided into TBs. Each TB can be 32×32, 16×16, 8×8, or 4×4 in size. The condition of CBF is shown in equation (3.8).

$$CBF = \begin{cases} 0, & if\ all\ coefficients\ in\ TB\ are\ zero \\ 1, & otherwise \end{cases} \tag{3.8}$$

Here the time required for encoding is saved by checking the coefficients of the quantized block. The coefficients are checked by using the CBF. If the block has all zeros, then the coding of that block can be skipped, which saves encoding time.
Let 'earlycu' is the variable used for the determination of the CU early. The 'earlycu' condition is checked by using the equation (3.9).

$$earlycu = \begin{cases} True, & if\ skip(0)\ is\ high \\ false, & otherwise \end{cases} \tag{3.9}$$

Here' skip (0)' checks the skip flag of the luma component. If the skip flag of the luma component is skipped, it returns true, which means the block is skipped. The advantage of this process is for the skipped CUs, the splitting and finding of RD cost can be avoided, which results in a decrease in encoding time.

3.3.3 Search Patterns

Two search patterns are proposed: New Cross Diamond Diamond (NCDD) and New Cross Diamond Hexagonal (NCDH) search patterns. In these search patterns, center biased searching is used and also allows halfway search stop. The search patterns are explained below.

New Cross Diamond Diamond (NCDD) search and New Cross Diamond Hexagonal (NCDH) search

In the MLRVS algorithm, the NCDH search pattern is used, which is formed by adding the third stage to the cross diamond hexagonal search [86], as shown in Fig. 3.8. The NCDD and NCDH patterns shown in Fig. 3.7 and Fig. 3.8 are explained below.

Step1 Perform a small diamond search by considering the median predictor as an origin (0, 0). Here four points around the origin are considered, with distance one for finding the best motion vector. The four search points are indicated by '•.' If the best

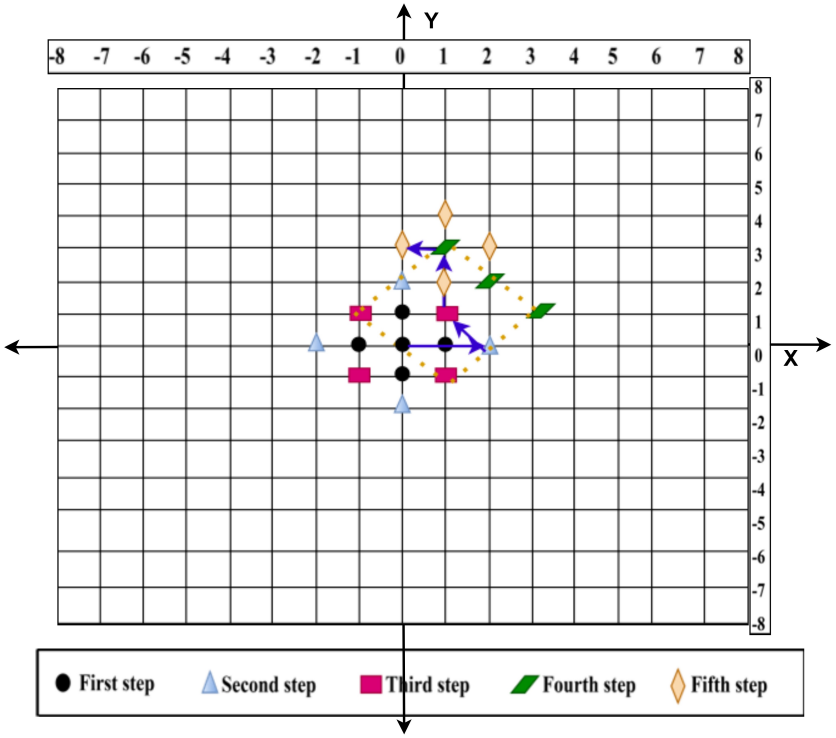


Figure 3.7 NCDD search pattern

motion vector is the same as the origin, then the searching process stops; otherwise, move to step2.

Step2 Again, consider the median predictor and make it an origin. Now take the four search points indicated by 'Δ' with a distance of two around the origin. If the best motion vector after searching is the same as the origin, the search stops; otherwise, move to step 3.

Step3 Now consider the two nearby search points indicated by '□' close to the best motion vector of step 2. Here the best motion vector can be found among the three search points, including the best motion vector of step 2.

Step4 (a) If the search pattern is NCDD, then the eight-point diamond search is applied by considering the best motion vector of step 3 as a center. If the obtained motion vector after searching is the same as the center, stop the searching operation. Otherwise, move to step 5.

(b) If the search pattern is NCDH, then the six-point hexagonal search is applied by considering the best motion vector of step 3 as a center. Stop the searching process

if the center point has a minimum distortion value. Otherwise, move to step 5.

Step5 Perform a small diamond search (like step 1) by considering the obtained motion vector as the center. The point with minimum SAD value is the best matching point.

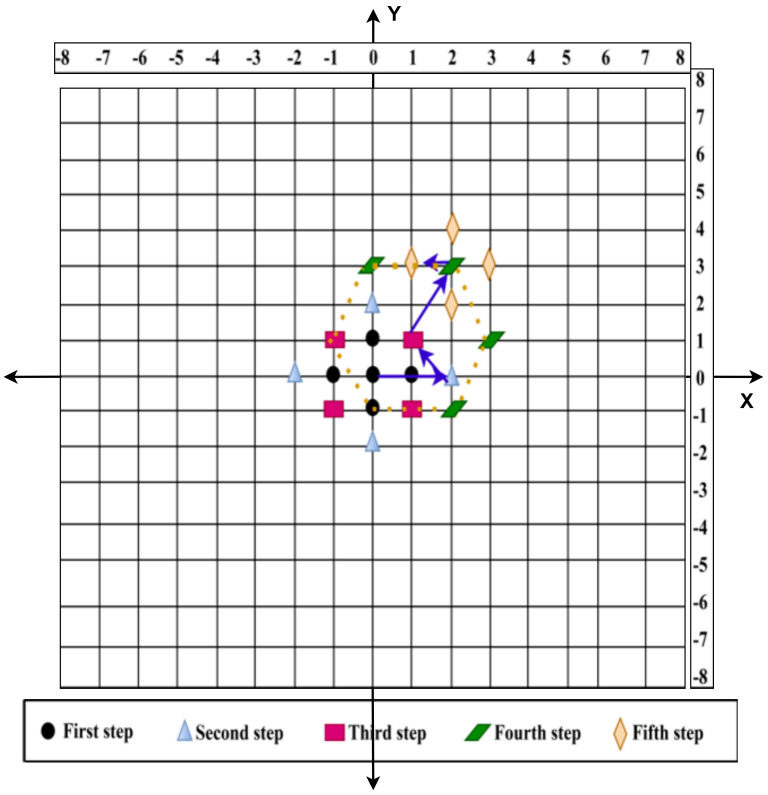


Figure 3.8 NCDH search pattern

3.4 Experimental Results

The proposed method is evaluated using the HEVC reference software HM 16.5 [87]. Seventeen different sequences with different resolutions are used to evaluate the output of the proposed method. We also measured the RD performance loss of the proposed algorithm using the Bjontegaard delta bitrate (BD-BR) [88, 89] and compared it with the state-of-the-art techniques in [49], [90] and [35]. Table 3.1 show the experimental conditions needed to verify the performance of the MLRVS algorithm. The percentage of Time Saving (TS) can be calculated using equation (3.10).

$$Timesaving(TS)(\%) = \frac{T_{orig} - T_{prop}}{T_{orig}} \times 100 \tag{3.10}$$

With the fast encoding options, the conventional HM reference software HM 16.5 is used as an anchor method.

Table 3.1 Experimental conditions

Maximum CU and TU size	64×64 and 32×32
Configuration	Encoder_randomaccess_main
QP values	22, 27, 32, and 37
Maximum CU and TU depth	4 and 3
GOP Size	8
Search range	64
Number of frames to be encoded	100

As discussed before, the main objective of the proposed method is to minimize the encoding time. The proposed method using each search pattern i.e., NCDD or NCDH is simulated separately and analyzed the results. Table 3.2 compares the proposed method using the NCDD search pattern (Prop +NCDD) with the standard HM 16.5. The findings indicate that the encoding time is reduced by 55% at the cost of a 0.31dB decrease in YPSNR and an 8.06% increase in bit rate. The proposed method using NCDH (Prop+NCDH) search pattern is also simulated and compared with HM 16.5 method. The outcome shows that the approach proposed significantly decreased the encoding time by 56% with minimal video quality degradation, i.e., 0.23 dB.

Table 3.3 compares the proposed methods (Prop+NCDD and Prop+NCDH) with [49] and [90] by making HM 16.5 reference method as an anchor. The authors in [49] reduced the encoding time by 32% using the early skip mode decision with slight RD performance loss. The results show the complete domination of the proposed method compared to [49] in encoding timesaving. Even though the bit rate is increased, the proposed method's timesaving percentage is almost 40% more than [49].

The authors in [90] use the machine learning approach to reduce the encoding time for finding the CU size. This method reduces the encoding time by 49% on average with

Table 3.2 Experimental outcomes of the proposed method compared to HM-16.5 standard

Class	Input	Resolution	Prop+NCDD							Prop+NCDH						
			BD-BR	BD-PSNR	TS (%)					BD-BR	BD-PSNR	TS (%)				
			(%)	(dB)	QP=22	QP=27	QP=32	QP=37	Total	(%)	(dB)	QP=22	QP=27	QP=32	QP=37	Total
A	PeopleOnStreet	2560x1600	7.83	-0.51	54.61	58.81	63.92	66.54	61	6.53	-0.26	60.48	62.90	66.59	70.12	65
	Traffic		8.98	-0.37	57.18	63.01	67.94	70.78	65	8.64	-0.28	58.62	61.97	66.52	71.12	65
B	Cactus	1920×1080	11.03	-0.27	34.17	51.09	60.55	68.42	54	9.94	-0.28	34.88	48.06	61.59	68.66	53
	Kimono		9.68	-0.23	39.80	52.50	67.88	76.81	59	7.07	-0.18	41.12	51.54	67.77	76.97	59
	Parkscene		6.92	-0.15	30.47	51.34	69.12	74.58	56	4.18	-0.09	26.05	48.90	65.16	75.22	54
	BasketballDrive		7.79	-0.42	35.40	40.85	45.55	59.51	45	5.26	-0.34	36.54	46.18	53.39	57.99	49
C	BasketballDrill	832×480	10.14	-0.40	27.34	35.38	45.87	54.32	41	12.07	-0.49	32.65	42.63	51.45	61.73	47
	BQMall		13.44	-0.53	32.76	42.66	54.03	58.28	47	10.45	-0.43	26.46	39.74	50.79	55.50	43
	PartyScene		7.05	-0.34	29.13	34.86	48.59	61.07	43	6.78	-0.14	23.92	31.05	41.58	56.28	38
D	BlowingBubbles	416×240	9.26	-0.36	22.10	32.88	42.60	52.15	37	4.50	-0.18	25.23	33.86	43.90	53.80	39
	BQSquare		8.84	-0.42	36.99	47.71	59.22	68.61	53	4.15	-0.10	33.90	53.37	66.87	76.99	58
	BasketballPass		7.36	-0.35	36.56	41.68	54.42	63.49	49	6.87	-0.33	43.01	50.01	59.02	73.33	56
	RaceHorses		10.80	-0.54	29.72	36.96	45.31	57.82	42	9.34	-0.46	29.77	43.20	51.54	63.98	47
E	KristenAndSara	1280×720	3.39	-0.10	56.57	68.08	73.57	77.26	69	2.90	-0.06	56.98	68.01	75.21	76.61	69
	Johnny		5.35	-0.12	61.13	70.85	76.24	78.36	72	2.53	-0.13	50.80	68.29	73.91	76.98	67
	FourPeople		4.17	-0.15	57.98	70.77	75.92	78.35	71	2.45	-0.09	55.71	65.43	67.69	74.78	66
	Stockholm		5.13	-0.14	52.10	60.32	68.41	71.31	63	4.86	-0.17	52.11	62.25	66.74	72.20	63
Average			**8.06**	**-0.31**	**40.82**	**50.57**	**59.94**	**66.92**	**55**	**6.38**	**-0.23**	**40.48**	**51.60**	**60.56**	**68.36**	**56**

Table 3.3 Experimental findings of the proposed method and state-of-art techniques using HM-16.5 as an anchor

Class	Input	Resolution	Prop+NCDD			Prop+NCDH			[49]			[90]			[35]		
			BD-BR	BD-PSNR	TS	BD-BR	BD-PSNR	TS	BD-BR	BD-PSNR	TS	BD-BR	BD-PSNR	TS	BD-BR	BD-PSNR	TS
			(%)	(dB)	(%)	(%)	(dB)	(%)	(%)	(dB)	(%)	(%)	(dB)	(%)	(%)	(dB)	(%)
A	PeopleOnStreet	2560x1600	7.83	-0.51	61	6.53	-0.26	**65**	**3.71**	-0.24	35	7.37	-0.38	56	**1.89**	**-0.21**	53
	Traffic		8.98	-0.37	**65**	**8.64**	-0.28	60	5.13	-0.26	40	9.07	-0.47	52	**2.01**	**-0.19**	49
B	Cactus	1920×1080	11.03	-0.27	**54**	**9.94**	-0.28	53	6.30	-0.19	32	7.53	-0.24	44	**1.67**	**-0.14**	28
	Kimono		9.68	-0.23	**59**	**7.07**	-0.18	**59**	**4.20**	-0.18	29	6.53	-0.27	51	**1.34**	**-0.16**	41
	Parkscene		6.92	-0.15	**56**	**4.18**	**-0.09**	**54**	5.40	-0.22	31	3.63	-0.14	53	**2.12**	**-0.09**	43
	BasketballDrive		7.79	-0.42	45	5.26	-0.34	**49**	**6.20**	**-0.09**	**24**	6.34	-0.15	46	**1.30**	-0.12	41
C	BasketballDrill	832×480	10.14	-0.40	41	12.07	-0.49	47	6.80	-0.21	35	9.81	-0.43	54	**1.92**	**-0.07**	34
	BQMall		13.44	-0.53	**47**	**10.45**	-0.43	43	5.60	-0.22	30	9.64	-0.48	42	**1.46**	**-0.10**	32
	PartyScene		7.05	-0.34	43	6.78	**-0.14**	**38**	3.92	-0.16	32	9.87	-0.76	59	**2.32**	-0.17	**35**
D	BlowingBubbles	416×240	9.26	-0.36	37	4.50	-0.18	**39**	**5.42**	**-0.09**	**26**	6.17	-0.37	37	**1.50**	-0.12	30
	BQSquare		8.84	-0.42	53	4.15	-0.10	**58**	**4.20**	-0.12	31	12.34	-0.87	47	**0.98**	**-0.06**	54
	BasketballPass		7.36	-0.35	49	6.87	-0.33	**56**	**2.30**	-0.18	25	10.05	-0.54	40	**0.54**	**-0.08**	51
	RaceHorses		10.80	-0.54	42	9.34	-0.46	47	5.10	**-0.11**	**33**	12.89	-0.8	57	**1.23**	-0.13	56
E	KristenAndSara	1280×720	3.39	-0.10	**69**	**2.90**	**-0.06**	**69**	**2.50**	-0.07	38	13.35	-0.62	58	**2.15**	**-0.06**	54
	Johnny		5.35	-0.12	**72**	**2.53**	**-0.13**	67	3.50	**-0.10**	**28**	8.09	-0.32	47	2.70	-0.11	57
	FourPeople		4.17	-0.15	**71**	**2.45**	**-0.09**	**66**	2.70	-0.13	32	9.07	-0.48	36	2.61	-0.12	62
	Stockholm		5.13	-0.14	63	4.86	-0.17	**63**	**2.60**	**-0.05**	**39**	8.44	-0.49	52	**1.94**	-0.07	61
Average			8.06	-0.31	55	6.38	-0.23	**56**	**4.44**	-0.15	32	8.83	-0.45	49	**1.73**	**-0.11**	**47**

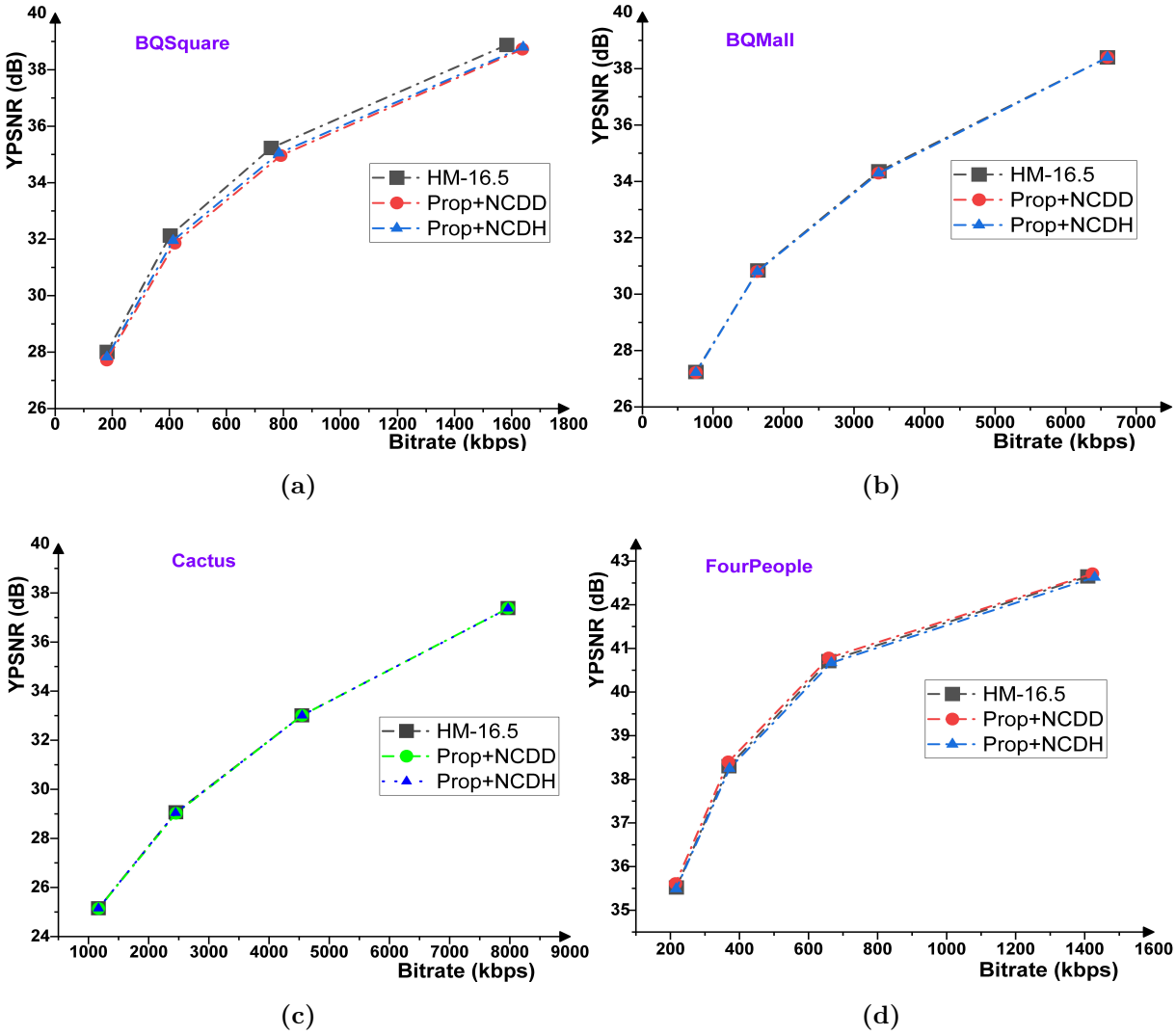


Figure 3.9 Example RD curves of (a) BQMall (b) Cactus (c) FourPeople (d) BQSquare video sequences

0.45dB loss in video quality and an 8.83% rise in bit rate. For a few video sequences like RaceHorses, BasketballDrill, and PartyScene, the approach in [90] saves more encoding time than our proposed method. The proposed method outperformed the [90] method for the remaining video sequences in timesaving, bit rate, and YPSNR. The proposed method can obtain more encoding time saving than the machine learning approach without sacrificing much coding quality. We have also compared the performance of the proposed method with the [35] approach. The experimental findings show that the state-of-the-art method achieved good RD performance. However, only 47% of encoding time saved, which is less compared to our proposed method.

The Peak Signal to Noise Ratio (PSNR) is calculated using the equation(3.11).

$$PSNR = 10log_{10}\frac{(2^{bitdepth}-1)^2 * W * H}{\sum_i (O_i - D_i)^2} \tag{3.11}$$

Where

bitdepth = each pixel bit depth

W = Number of horizontal pixels

H = Number of vertical pixels

O_i= reference picture pixel value

D_i= Decoded picture pixel value

i = pixel address

As human vision is more sensitive to luminance (Y), the YPSNR is considered instead of PSNR for drawing the RD curve. Fig. 3.9 shows an example of RD curves for BQSquare, BQMall, Cactus, and FourPeople, respectively. The RD curves indicate that the proposed approach can maintain the video's quality the same as that of the regular HM 16.5. The RD performance loss due to the proposed method is slightly larger than standard HM but tolerable and even smaller than the machine learning approach method. The encoder_randomaccess_main configuration is used, which uses the hierarchical Bidirectional structures. This configuration provides higher efficiency but with a more significant delay compared to the other configurations.

3.5 Conclusions

The MLRVS algorithm is used to minimize the encoding time of the HEVC encoder. This algorithm uses vertical subsampling, which decreases the number of computations needed to find the motion vector. Two search patterns are proposed, which helps to quicken the motion estimation process. Moreover, the complexity reduction algorithm is used to lessen the time required for coding the coefficients. The proposed algorithm with two different search patterns is simulated individually. The results exhibit that the proposed algorithm has reduced the encoding time by 56% with NCDH and 55% with NCDD search patterns compared to the HM 16.5 standard. The results exhibit that our proposed method saves more encoding time than the state-of-the-art methods with slight RD performance loss.

Chapter 4

Fast Convolutional Neural Network based Coding Unit Size Prediction in HEVC

4.1 Introduction

The High Efficiency Video Coding (HEVC) provides compression for video sequences with better video quality at lower bit-rates compared to H.264/Advanced Video Coding (AVC) standard. Advanced coding methods, such as the quad-tree Coding Tree Unit (CTU) partition structure, improve the efficiency of HEVC. However, because of the recurrent Rate-Distortion Optimization (RDO) search procedure, CTU partitioning takes longer. The RDO search mechanism and motion estimation accounts for 80% of the entire encoding time. The complexity in determining the Coding Unit (CU) size can be lowered to reduce encoding time. The encoding time can be further reduced by improving the speed of the motion estimation process.

The motion estimation process helps in finding the motion vector to predict the next frame. It also helps in reducing the temporal redundancy between the current frame and the reference frame. The frame at time 't' is considered the current frame, and the frame at 't-1' is taken as a reference frame. The motion estimation process determines the motion vector by finding the matching block in reference frame for each block in the current frame. The motion vector represents the displacement between the position of the block in the reference frame and the corresponding block position in the current frame. Several search patterns like a Full search, Logarithmic search, a Three-step search,

Hexagonal search [91], etc., can be used to find the motion vectors at a less computational expense.

The HEVC employs a quad-tree structure to divide the frame into Coding Tree Units (CTUs). The CTU with optimal CU partitions is considered for encoding in HEVC. The CU partitions with minimal RD cost are treated as the optimal CU partitions determined using the Rate-Distortion Optimization (RDO) search process. The optimal CTU partitions of the BasketballPass sequence are shown in Fig. 4.1. The CU partitions show that the smaller CUs contain more information than the large CUs. The process of determining the optimal CUs consumes more time due to the conventional RDO search process. Some of the techniques to determine the optimal CTU partitions are provided below.

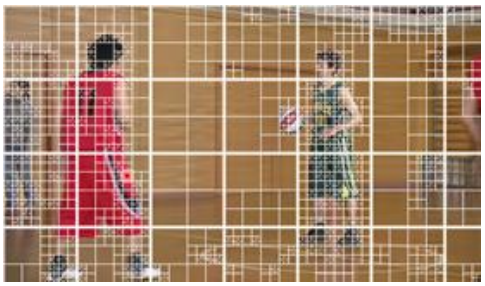


Figure 4.1 Optimal CTU partitions of BasketballPass sequence

The authors in [92] proposed a method that determines the best PU mode by applying the zero residual quad-tree process. Hyo-Song Kim and Rae-Hong Park [93] proposed an algorithm, which determines the CU partitions fastly based on the Bayesian decision rule. The authors in [94] reduced the encoding time by pruning the quad-tree based on residual statistics obtained after prediction. Leng et al. [95] proposed a method that decides the CU depth by skipping the rarely used CU depths of the past frames. Shen et al. [96] proposed an approach that determines CU's splitting based on RD-cost and prediction error. Cho et al. [97] use the Bayesian rule to decide the CU split based on the features of low-complexity RD costs. Yoo et al. [98] determine the partition of PU having maximum probability by considering the RD cost and coded block information of already encoded PUs. Correa et al. [99] use data mining techniques with early termination schemes to decide CU's partition structure.

In [100], the authors proposed a method to predict the PU mode and CU partition

using the binary and multi-class SVM classifier. Helen et al. [101] use the neural network for the analysis of CTU depth. The authors also predict the CU size using the deep learning approach. The combination of the depth analysis method and the CU prediction approach helps in decreasing the complexity of the encoder. Tianyi et al. [102] use the deep learning CNN approach instead of the RDO process to predict the CTU partition in intra mode HEVC. Mai et al. [103] use the Long and Short Term Memory (LSTM) approach that reduces the complexity in HEVC for both intra and inter modes. The authors in [104, 105] use the Asymmetric Kernel CNN (AK-CNN) approach that lessens the time required for CU/PU size determination. Zhenyu et al. [106] reduced the VLSI hardware encoder's complexity by using the CNN approach, which is used to predict the CU partitions. The above works concentrate on reducing the encoding time by predicting the optimal CU partitions fastly. However, the motion estimation increases the encoding time significantly.

We proposed the deep learning Convolutional Neural Network (CNN) approach for CU size prediction and Multi-Resolution frame with the Cross Diamond Octagonal search pattern (MRCDO) for motion estimation to reduce the encoding time. The combination of the prediction approach and motion estimation method reduced the encoding time by 66.91%.

4.2 Rate-Distortion Optimization (RDO) search process in HEVC

In HEVC, each frame is divided into Coding Tree Units (CTUs). The maximum size of CTU is 64×64, and the CTU can be recursively split into four equal-sized Coding Units (CUs). The CU can have sizes of 64×64, 32×32, 16×16, and 8×8 at the depths of 0, 1, 2, and 3, respectively. The CU size is determined using the RDO search process, which uses the top-down checking approach, as shown in Fig. 4.2. Consider the CU (Y), which is of size 64×64. The split conditions at different depths are given below
At depth 0, $J^{Y} \geq \sum_{i=1}^{4} J^{Y_i}$,
At depth 1, $J^{Y_i} \geq \sum_{i,j=1}^{4} J^{Y_{i,j}}$ and
At depth 2, $J^{Y_{i,j}} \geq \sum_{i,j,k=1}^{4} J^{Y_{i,j,k}}$
Where, $i, j, k \in \{1, 2, 3, 4\}$ are the subscripts and J^{Y} represents the RD cost of CU (Y). Y,

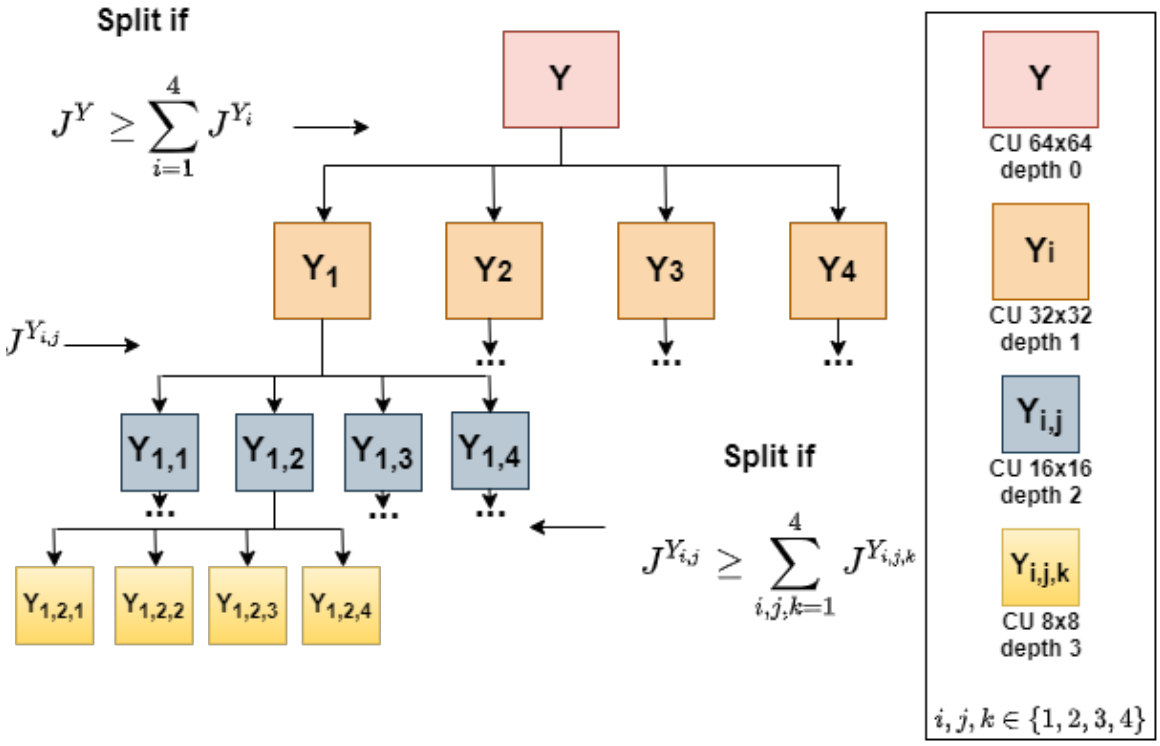


Figure 4.2 Three level classifier structure with split condition

Y_i, $Y_{i,j}$ and $Y_{i,j,k}$ are the CUs present at depth0, depth1, depth2 and depth 3, respectively. The RD cost (J) is calculated by using equation (4.1).

$$J = D_{SAD} + \lambda \times R \tag{4.1}$$

Where, λ= Lagrangian multiplier, R = number of bits needed to transmit and D_{SAD} = Distortion.

At depth 0, if the split condition is true, split Y and represent the sub-CUs at depth 1 as $\left\{Y_i\right\}_{i=1}^{4}$. Then the each CU at the depth 1 is furthur split into four equal sized sub CUs, if the split condition is satisfied at depth 1. $\left\{Y_{i,j}\right\}_{i,j=1}^{4}$ represents the CUs at depth 2. Finally, the third condition decides whether the CUs at depth 2 needs to be split or not. If the condition is true, $\left\{Y_{i,j}\right\}_{i,j=1}^{4}$ is divided into four sub-CUs $\left\{Y_{i,j,k}\right\}_{i,j,k=1}^{4}$.

The RD cost calculation and split condition checking process consumes more time in standard HEVC. For 64×64 CTU, 85 CUs (if the whole CTU is divided into 8×8 CUs) need to be checked, including N×N size CUs. The 'N' value can be 8, 16, 32, or 64. The pre-coding of 85 CUs consumes more time in standard HEVC. The accurate prediction of CU size can reduce the encoding time significantly.

To reduce the encoding time, we proposed the CU size prediction using CNN and the MRCDO motion estimation method.

4.3 Proposed work

The HEVC encoding process involves Rate-Distortion Optimization, which determines the CU size. As the RDO process consumes more time, we presented a deep learning CU size prediction approach using CNN. The MRCDO method using the Cross Diamond Octagonal (CDO) search pattern is also proposed to determine the motion vector quickly. The CU size prediction and MRCDO method are discussed below.

4.3.1 MRCDO search method

The MRCDO search algorithm is a MLRVS motion estimation algorithm (discussed in section 3.3.1) with CDO search pattern. The CDO pattern is used as a search pattern to find motion vectors. The CDO search pattern illustrated in Fig. 4.3 is described below.

Step 1: Apply four-point diamond search pattern by considering the median predictor as starting point i.e., origin (0, 0). The four points are indicated by '•.' Determine the point which has the smallest SAD value.

Step 2: If the origin has a minimum SAD value, terminate the search operation. Otherwise, move to step3.

Step 3: Now apply large diamond search pattern around the origin. Perform SAD computations for the search points applied around the origin, which are indicated by '▲.'

Step 4: If the origin has a minimum SAD value, stop the search operation, otherwise proceed to Step 5.

Step 5: Consider the two surrounding points denoted by '■' to the minimum SAD value point in step 3 and perform SAD computations for these two search points.

Step 6: Now consider the search point in the previous step, which has the lowest SAD value as the center and apply the octagonal pattern around it. The 'parallelogram' symbol represents the octagonal search points.

Step 7: Consider the previous step minimum SAD value point as center and apply the four points as shown in step 1. The point with the lowest SAD value is considered the best match.

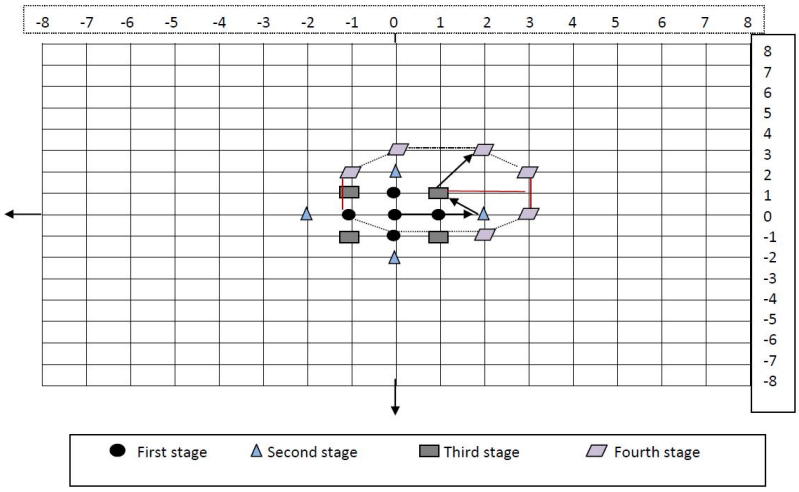


Figure 4.3 Cross Diamond Octagonal search pattern

The TZ search algorithm with search patterns aids in motion estimation complexity reduction. However, the search patterns might get trapped to local minima, which leads to performance loss. To address the local minima problem, the MRCDO search method is developed, which increases the probability of obtaining global minima.

4.3.2 CU size prediction using CNN

The CNN model depicted in Fig. 4.4 is trained over three CU depth levels using the database to derive the CTU partitions. The database consists of 397 video files, out of which 18 sequences from the Joint Collaborative Team on Video Coding (JCT-VC) test set, 300 videos from Stanford university, and 79 sequences from Xiph.org. The videos belong to different sizes such as (112×112), (352×288), (352×240), (416×240), (832×480), (1280×720), and 1080p. The CU depth data is generated by encoding the aforementioned sequences at four QPs using HEVC reference software. In addition, with

the random_access_main configuration, 19,607,566 samples were collected. The loss and accuracy curves for training and validation data at four different Quantization Parameter (QP) values are shown in Fig. 4.5.

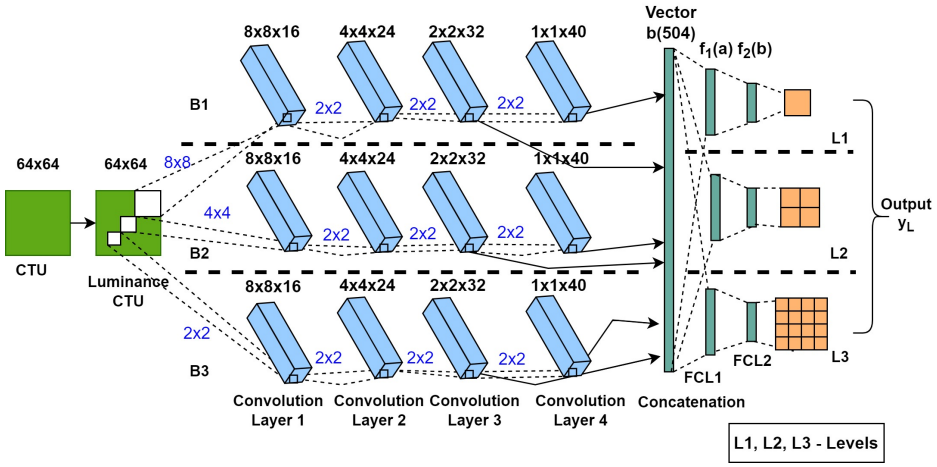
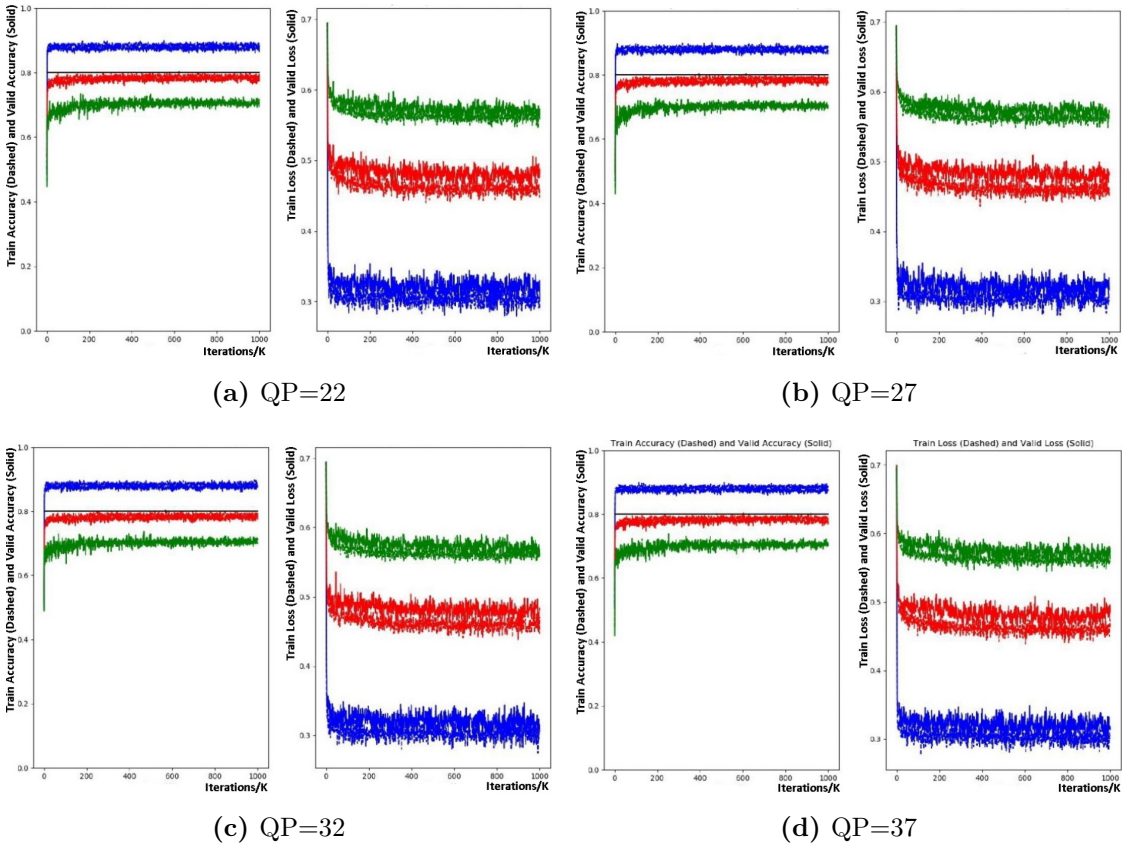


Figure 4.4 CU size prediction using CNN

The CU depth levels are represented with L, where L belongs to 1, 2, and 3. The CU partitions are made at three different levels based on the values of Y1, Y2, and Y3 values. The average accuracy for predicting the CU partitions at level 1, level 2, and level 3 for inter mode HEVC are 86.36%, 82.18%, and 77.32%, respectively.

The video sequence is the sequence of frames, where each frame is segmented into CTUs. We provide the luminance of CTU, which is of 64×64 size as an input to the CNN. The elements of the luminance CTU undergo normalization, which results in all the values of CTU are [0, 1]. The resultant CTU undergoes down-sampling that results in CU with sizes 16×16, 32×32, and 64×64, respectively. The CNN model consists of four convolution layers, a concatenation layer, and two fully connected layers.
The working of each layer in the CNN is discussed below.

1. **Convolution layers** There are four convolution layers in the CNN. At the convolution layer 1, 16 filters with a kernel size of 8×8,4×4, and 2×2 are applied at three branches B1, B2, and B3, to obtain the features. The three branches resultant features are individually convolved with 2×2 kernel at convolution layer2, layer3 and layer4 to extract low-level features. Layer2, layer3 and layer 4 have 24, 32 and 40 filters, respectively. Note that the kernel size is taken as stride length.



(a) QP=22 **(b)** QP=27

(c) QP=32 **(d)** QP=37

Figure 4.5 Loss and accuracy curves for training and validation data of the CNN model

2. **Concatenation layer and Fully Connected Layers (FCL)** At the concatenation layer, the feature maps obtained at the output of the last two convolution layers at three branches are combined to generate a single vector. The output vector is represented by 'b,' which is of size 504 features. Then the output vector is passed through two FCLs, and finally, the output is predicted at the output layer. The f1(a) and f2(b) represent the features of FCL1 and FCL2.

Fig. 4.6 shows the CU classification based on the y_L output. At level 1, if y_1 is one, then proceed to the prediction at level 2. The calculation of feature vectors f1(a) and f2(b) of the fully connected layers is performed at this level. Based on the resultant feature vectors, the output y_2 is decided. If the output y_2 is zero, stop the feature vector calculation of fully connected layers at level 3. Otherwise, perform prediction at level 3 by calculating the f1(a) and f2(b) feature vectors of FCL1 and FCL2 to predict the output. The convolution and FCLs are activated with Rectified Linear Units (ReLU), and the output layer is activated with sigmoid function.

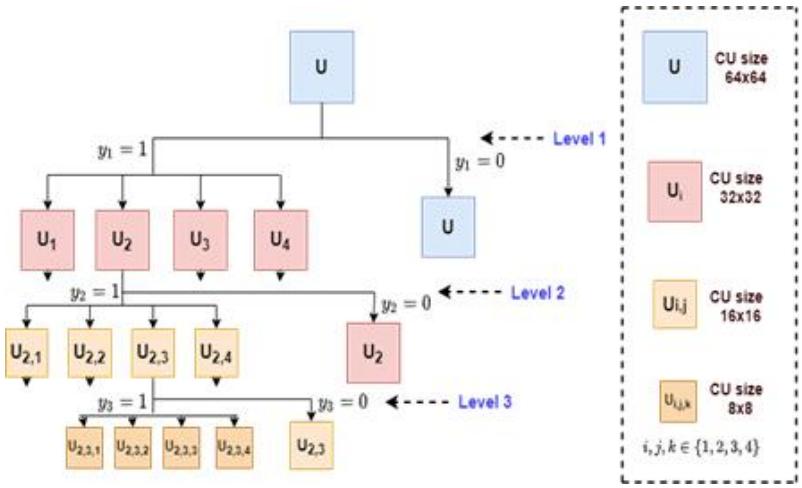


Figure 4.6 CU classification based on y_L output, L represents the Level

During the CNN training, the features at the output of the second fully connected layer are dropped with a 50% probability. The output y_L is binary, which decides whether the CU is to be split or not.

Equation (4.2) shows the CU size based on the y_L output.

$$y_L = \begin{cases} 32 \times 32\, CUs, & L = 1 \\ 16 \times 16\, CUs, & L = 2 \\ 8 \times 8\, CUs, & L = 3 \end{cases} \tag{4.2}$$

CNN delivers the probability of binary output at each level when CTU is given as an input. The maximum and lower limits at level 1 are 0.6 and 0.4, respectively. Similarly, the upper and lower threshold for level 2 are 0.7 and 0.3, and for level 3, they are 0.8 and 0.2. The CU is split at each level based on the threshold value.

Consider there are P training samples, the loss function L_f using cross-entropy is given as

$$L_f = -\frac{1}{r}\sum_{r=1}^{r}[Y_L^{(r)} \log \hat{Y_L}^{(r)} + (1 - Y_L^{(r)}) \log(1 - \hat{Y_L}^{(r)})] \tag{4.3}$$

Where,

$Y_L^{(r)}$ = ground-truth value of the r_{th} training sample,

$\hat{Y_L}^{(r)}$ = corresponding prediction to $Y_L^{(r)}$

During training, the stochastic gradient descent algorithm is used to optimize the loss L_f.

4.4 Experimental Results

In this section, we evaluate the proposed approach's performance using HM 16.5 reference software. The encoder_random_access_main configuration is used to evaluate the proposed method. The 64-bit Windows-7 operating system with Intel(R) Core(TM) i5 is used for our experiments. Here, the QP values of 22, 27, 32, and 37 compress the video sequences. The RD performance of the proposed method is analyzed in terms of Bjontegaard delta PSNR (BD-PSNR) and Bjontegaard delta bit-rate (BD-BR). Another metric, Time Saving (TS), is used to measure the percentage of time-saving compared to the standard HM 16.5. The percentage of time-saving is measured by using equation (4.4).

$$TimeSaving(TS)(\%) = \frac{T_{orig} - T_{prop}}{T_{orig}} \times 100 \tag{4.4}$$

Initially, we have analyzed the performance of the MRCDO method in comparison to HM 16.5. Table 4.1 shows the comparison of MRCDO and HM 16.5 methods. It is observed that the MRCDO method has saved the encoding time by an average value of 54.26%. We can also note that an average value of 7.76% rise in BD-BR and 0.30dB decrease in BD-PSNR compared to the HM 16.5.

Then we analyze the proposed approach, which predicts the CU size using CNN in combination with the MRCDO method. Our proposed method is compared with three state-of-art methods by making the HM as an anchor. The HM uses the conventional RDO search in combination with the TZ search algorithm.

Table 4.2 results show that the proposed method saves the encoding time by 66%, 64.99%, 67.20%, and 69.46% on average, outperforming the 59.34%, 57.98%, 55.59% and 57.87% time savings in [107] and 51.46%, 52.91%, 52.49% and 55.28% time savings in [90] at QP=22, 27, 32 and 37, respectively. In total, our proposed method saves the encoding time by an average of 66.91% compared to [107] @ 57.69% and [90] @ 53.03%.

The proposed method is also compared to another state-of-the-art method in [103]. The average saving of encoding time by [103] is 59.70%, which is lower than the proposed approach. The time savings of our proposed method is at the cost of RD performance loss. Table 4.2 shows that the average rise in BD-BR is 9.16%, which is better than [107]. But compared to [90], the increase in BD-BR is slightly higher. Similarly, the average decrease

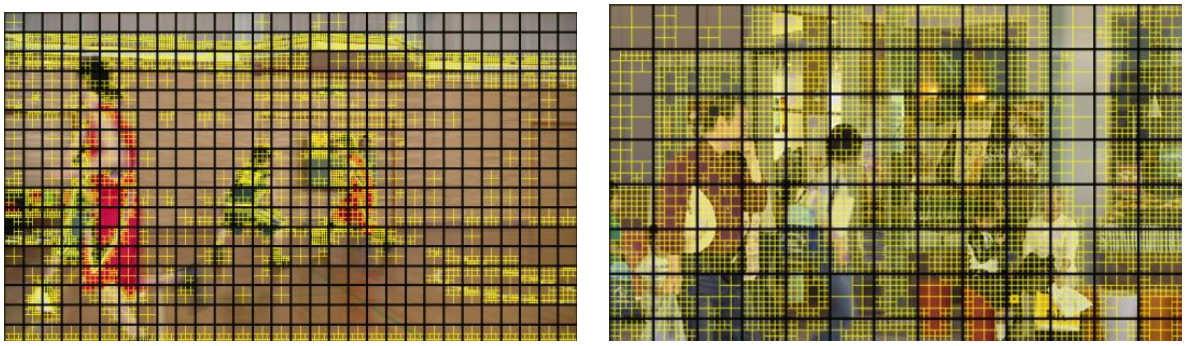
Table 4.1 Comparison results of MRCDO and HM-16.5

Class	Size	Video Sequences	BD-BR (%)	BD-PSNR (dB)	TS(%)				
					QP=22	QP=27	QP=32	QP=37	Tot
A	2560×1600	PeopleOnStreet	5.22	-0.14	56.27	58.69	62.38	65.91	60.81
		Traffic	7.33	-0.18	49.41	52.76	57.31	61.91	55.34
B	1920×1080	Kimono	5.22	-0.58	35.88	45.43	50.46	53.29	46.26
		ParkScene	7.51	-0.35	14.90	28.99	38.91	45.98	32.19
		Cactus	7.58	-0.17	44.01	50.97	63.72	71.06	57.44
		BasketballDrive	10.43	-0.20	18.14	29.72	36.90	46.88	32.91
		BQTerrace	8.09	-0.21	23.84	41.73	60.99	70.99	49.38
C	832×480	BQMall	10.77	-0.43	35.46	46.74	60.71	67.89	52.70
		BasketballDrill	9.55	-0.12	28.10	62.23	69.27	77.05	59.16
		RaceHorses	10.95	-0.46	18.70	29.98	32.97	43.42	31.26
		PartyScene	6.23	-0.43	63.58	62.69	63.63	64.54	63.61
D	416×240	BasketballPass	7.43	-0.34	43.07	54.76	63.37	74.26	58.86
		BlowingBubbles	9.06	-0.35	29.06	36.19	51.78	61.39	44.60
		RaceHorses	11.27	-0.57	20.44	26.78	34.56	45.47	31.81
		BQSquare	10.48	-0.47	35.81	55.83	68.75	79.65	60.01
E	1280×720	FourPeople	4.20	-0.21	67.38	80.41	87.21	89.57	81.14
		KristenAndSara	3.24	-0.10	69.15	80.67	85.20	89.81	81.20
		Johnny	5.12	-0.14	63.06	77.38	84.56	87.42	78.10
Average			**7.76**	**-0.30**	**39.79**	**51.21**	**59.59**	**66.47**	**54.26**

(a) **(b)**

Figure 4.7 Reconstructed frames of (a) BasketballDrive (b) BQMall at QP=27

in BD-PSNR is 0.52dB, which is lower than 0.85dB in [107] and somewhat higher than 0.47dB in [90]. The proposed method outperforms [107] in terms of BD-BR, BD-PSNR,



(a) **(b)**

Figure 4.8 Reconstructed frames with predicted CTU partitions of (a) BasketballDrive (b) BQMall at QP=27

and TS. Our approach is also superior to [90] and [103] in terms of time-saving (TS).

Fig. 4.7(a) and (b) show the reconstructed frames of BasketballDrive and BQMall video sequences when QP=27. Fig. 4.8(a) and (b) display the reconstructed frames with CU partitions predicted by the proposed method. The black boxes in Fig. 4.8 represent the CTU, which is of size 64x64. The yellow colour boxes represent the CU partitions present inside the CTU. The BasketballDrive frame contains more CTU blocks with no CU partitions.

The CTU with no partitions represents that the output y1 at level 1 of CNN is zero. If y1 is zero, the prediction at level 2 and level 3 is skipped out, which results in saving encoding time. If the CTU block contains only four CU partitions, then the output y2 at level 2 is zero. If y2 is zero, the prediction is not performed at level 3. A large number of CU partitions results in more encoding time. The BQMall frame in Fig. 4.8(b) contains more CU partitions, resulting in more encoding time than the BasketballDrive sequence.

The example RD curves for four video sequences at four different video resolutions are shown in Fig. 4.9. The RD curves show that a slight decrease in YPSNR is observed compared to HM-16.5. For the four video sequences, the degradation in YPSNR is less than 1dB, which shows a negligible loss in video quality.

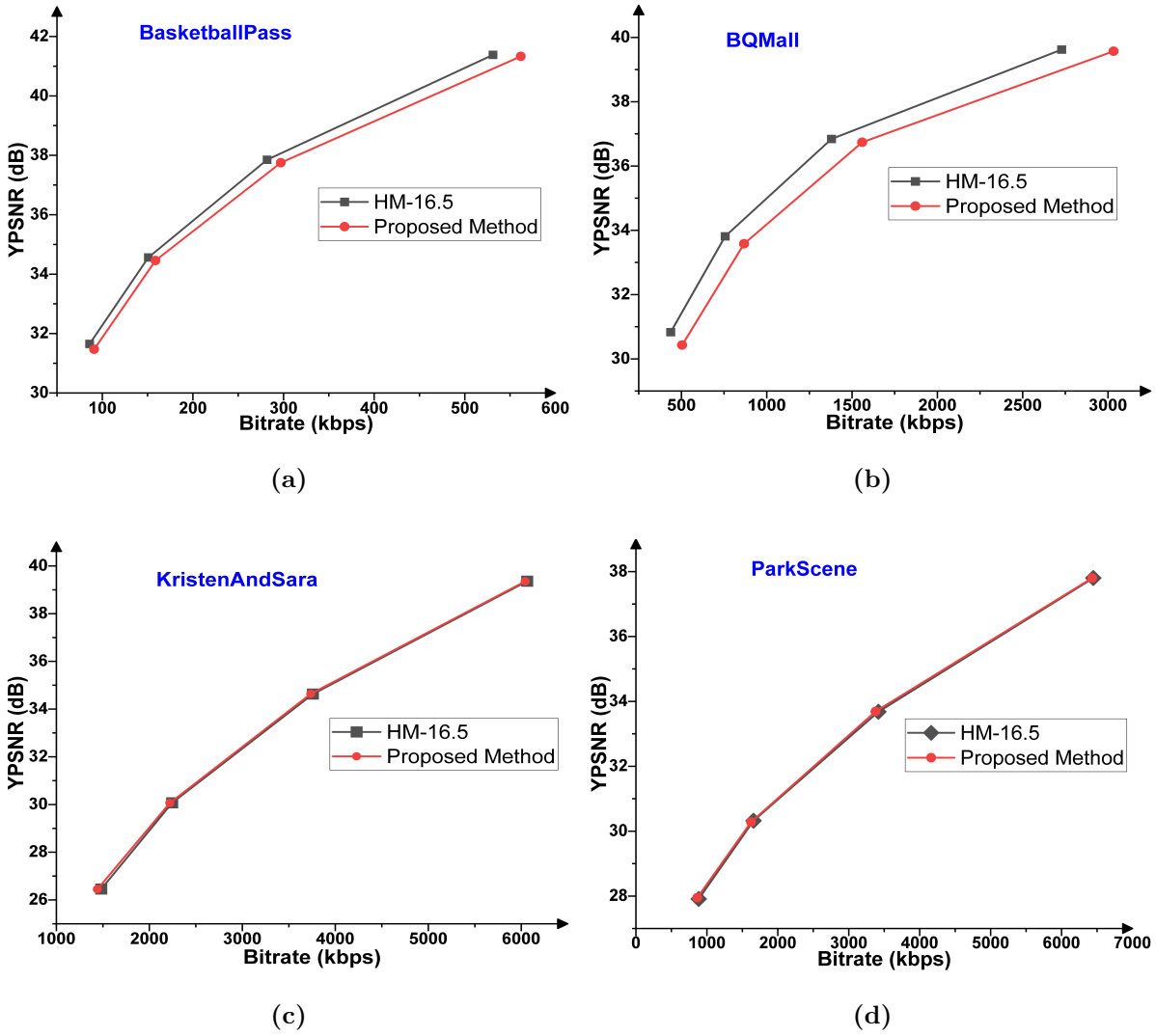


Figure 4.9 Example RD curves of (a) BasketballPass (b) BQMall (c) KristenAndSara (d) ParkScene

4.5 Conclusions

In HEVC, the CTU partition structure is determined using the RDO search procedure. The recursive RDO search process increases the encoding time. We suggested the CU size prediction method utilizing a deep learning methodology to reduce encoding time. The MRCDO search method is also proposed, which improves the motion estimation process speed. Compared to the HM-16.5 standard, the suggested solution reduces encoding time by 66.91 percent on average, with a 9.16 percent increase in BD-BR and a 0.52dB loss in BD-PSNR. In terms of both encoding time and RD performance, the suggested technique outperforms the state-of-the-art methods.

Table 4.2 Comparison results of proposed and state-of-the-art methods

Class	Video Sequence	Size	Approach	BD-BR (%)	BD-PSNR (dB)	TS(%)				
						QP 22	QP 27	QP 32	QP 37	Total
A	PeopleOnStreet	2560×1600	[107]	16.02	-0.86	63.39	58.99	57.01	56.96	59.08
			[90]	9.93	-0.42	52.96	54.75	57.83	62.56	57.02
			[103]	**2.31**	**-0.12**	57.41	60.43	59.61	62.03	59.87
			Our	6.78	-0.73	**70.19**	**67.84**	**71.19**	**69.06**	**69.57**
	Traffic		[107]	11.26	-0.58	60.84	60.25	58.94	60.16	60.04
			[90]	7.88	-0.53	54.64	58.88	62.83	64.92	60.31
			[103]	**2.63**	**-0.31**	53.61	57.81	61.23	67.55	60.05
			Our	7.85	-0.41	**73.14**	**69.10**	**76.25**	**75.42**	**73.47**
B	Kimono	1920×1080	[107]	10.41	-0.39	65.34	62.93	62.43	64.03	63.68
			[90]	8.49	-0.61	49.63	56.86	61.31	65.21	58.25
			[103]	**3.17**	**-0.26**	56.53	59.23	62.12	68.53	61.60
			Our	9.17	-0.67	**75.01**	**72.20**	**74.58**	**69.40**	**72.79**
	ParkScene		[107]	6.53	-0.27	61.48	60.90	60.59	62.12	61.27
			[90]	3.63	**-0.14**	41.69	44.79	59.98	64.92	52.84
			[103]	**2.12**	-0.31	54.40	57.41	61.30	69.71	60.70
			Our	7.92	-0.44	**67.84**	**75.05**	**71.91**	**74.48**	**72.32**
	Cactus		[107]	13.89	-0.50	60.59	58.89	58.06	58.60	59.03
			[90]	7.53	**-0.24**	38.37	40.83	43.61	51.23	43.51
			[103]	**2.77**	-0.29	55.41	58.74	61.21	63.53	59.72
			Our	9.37	-0.52	**73.58**	**65.78**	**71.83**	**74.56**	**71.43**
	BasketballDrive		[107]	11.62	-0.27	59.79	60.03	60.90	62.50	60.81
			[90]	9.01	-0.37	58.18	62.01	65.94	68.78	63.72
			[103]	**3.67**	**-0.19**	65.51	72.61	74.05	75.48	71.91
			Our	8.81	-0.21	**66.88**	**73.88**	**76.70**	**78.61**	**74.02**
	BQTerrace		[107]	13.05	-0.77	58.26	58.38	57.33	57.94	57.97
			[90]	5.76	-0.57	49.55	54.74	58.34	62.34	56.24
			[103]	**2.45**	**-0.23**	57.93	59.81	61.00	64.21	60.73
			Our	9.48	-0.32	**66.90**	**63.99**	**67.27**	**67.35**	**66.37**
C	BQMall	832×480	[107]	22.01	-1.27	57.25	54.39	55.40	56.71	55.93
			[90]	9.64	-0.48	52.62	42.97	35.52	37.12	42.05
			[103]	**2.92**	**-0.34**	52.41	57.32	64.81	66.72	60.31
			Our	11.83	-0.52	**64.81**	**65.03**	**65.36**	**69.37**	**66.14**
	BasketballDrill		[107]	22.48	-0.98	58.21	57.25	53.76	53.20	55.60
			[90]	9.81	-0.43	46.65	58.86	47.66	62.53	53.92
			[103]	**4.25**	**-0.42**	58.31	55.72	50.63	57.51	55.54
			Our	9.98	-0.88	**63.70**	**64.28**	**65.43**	**66.93**	**65.08**
	RaceHorses		[107]	12.89	-0.80	59.20	57.16	54.96	56.75	57.01
			[90]	6.78	-0.42	48.47	56.30	57.61	61.05	55.85
			[103]	**2.71**	**-0.18**	52.31	54.60	58.31	59.45	56.15
			Our	11.78	-0.53	**64.36**	**60.82**	**65.43**	**67.30**	**64.47**
	PartyScene		[107]	14.97	-0.95	**58.77**	56.49	48.97	54.01	54.56
			[90]	6.41	-0.40	50.76	**59.47**	**60.64**	**65.68**	**59.13**
			[103]	**2.21**	**-0.21**	54.69	48.52	55.40	51.97	52.64
			Our	8.36	-0.40	55.24	42.05	38.14	58.45	48.47
D	BasketballPass	416×240	[107]	18.35	-0.98	56.02	54.06	46.17	53.81	52.51
			[90]	10.05	-0.54	43.69	41.03	37.46	36.69	39.71
			[103]	**3.42**	**-0.32**	57.41	62.90	63.21	65.10	62.15
			Our	8.89	-0.74	**64.40**	**67.94**	**68.79**	**70.03**	**67.79**
	BlowingBubbles		[107]	13.99	-0.80	**57.26**	**54.88**	48.72	56.45	54.32
			[90]	6.17	-0.37	57.15	42.45	25.73	22.81	37.03
			[103]	**2.81**	**-0.27**	44.51	52.54	56.81	**60.54**	53.60
			Our	8.93	-0.57	54.76	53.28	**57.47**	57.90	**55.85**
	RaceHorses		[107]	17.08	-1.12	57.23	54.30	50.45	53.73	53.92
			[90]	8.49	-0.54	51.64	55.62	59.71	62.20	57.29
			[103]	**2.95**	**-0.18**	40.12	43.53	48.97	53.64	46.56
			Our	10.12	-0.46	**59.76**	**56.28**	**61.47**	**63.90**	**60.35**

Continued on next page

Table 4.2 – continued from previous page

Class	Video Sequence	Size	Approach	BD-BR (%)	BD-PSNR (dB)	TS(%)				
						QP 22	QP 27	QP 32	QP 37	Total
	BQSquare		[107]	21.55	-1.71	53.73	52.82	47.41	49.93	50.97
			[90]	12.34	-0.87	**61.45**	**62.40**	58.99	46.86	57.42
			[103]	**2.91**	**-0.24**	47.81	49.61	57.86	59.43	53.68
			Our	10.63	-0.73	54.30	58.39	**60.14**	**68.65**	**60.37**
E	FourPeople	1280×720	[107]	17.57	-0.96	59.49	57.58	58.01	58.67	58.43
			[90]	9.07	-0.48	53.52	40.88	26.12	24.34	36.21
			[103]	**3.71**	**-0.24**	61.46	59.64	60.61	61.12	60.70
			Our	8.28	-0.50	**67.02**	**67.59**	**69.87**	**70.91**	**68.84**
	KristenAndSara		[107]	24.36	-1.16	61.35	60.74	59.16	62.43	60.92
			[90]	13.35	-0.62	54.44	56.94	56.05	62.61	57.51
			[103]	**3.60**	**-0.20**	66.03	68.61	71.03	69.82	68.87
			Our	7.17	-0.37	**71.88**	**73.83**	**74.62**	**76.03**	**74.09**
	Johnny		[107]	23.09	-0.88	59.84	63.53	62.41	63.62	62.35
			[90]	7.37	-0.39	60.93	62.61	69.53	**73.15**	66.55
			[103]	**5.13**	**-0.27**	69.57	67.81	69.79	72.43	69.90
			Our	9.59	-0.33	**74.24**	**72.53**	**73.06**	71.88	**72.92**
Average			[107]	16.17	-0.85	59.34	57.98	55.59	57.87	57.69
			[90]	8.43	-0.47	51.46	52.91	52.49	55.28	53.03
			[103]	**3.10**	**-0.25**	55.86	58.16	61.00	63.82	59.70
			Our	9.16	-0.52	**66.00**	**64.99**	**67.20**	**69.46**	**66.91**

Chapter 5

CU Size Prediction in Scalable Video Coding using CNN-LSTM

5.1 Introduction

High Efficiency Video Coding (HEVC) is a standard that compresses Ultra High Definition (UHD) videos with a 50% less bit rate compared to the H.264 Advanced Video Coding (AVC) [108] standard while maintaining the same video quality. Considering the HEVC compression performance, the SHVC standard is developed. SHVC is the extension of HEVC, which is constructed using the fundamentals of HEVC. SHVC allows encoding the same video into various video resolutions, qualities, or frame rates. SHVC uses a Coding Unit, Prediction Unit(PU), and Transform Unit(TU) during the coding process. It consists of one Base Layer (BL) and multiple Enhancement Layers (ELs), shown in Fig. 5.1. SHVC can provide quality, temporal and spatial scalability by utilizing BL and EL. The BL represents the lowest quality video, and EL provides improved quality compared to BL.

In SHVC, the encoding starts at the Base layer and proceeds to the Enhancement Layers in the quality order. During the encoding process, the earlier frames from the same layer and the lower layer frames are used for the prediction. SHVC uses Inter-Layer Texture Prediction (ILTP) and Inter-Layer Motion Prediction (ILMP) to exploit the correlation between the motion vector and pixel values of non-identical layers.

ILTP sets the Inter-Layer Picture (ILP) as a reference frame for EL coding. The ILP

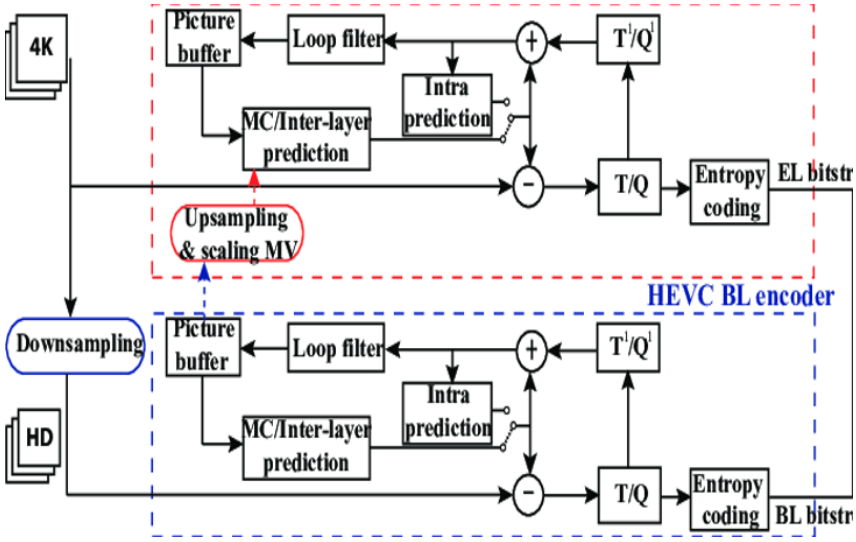


Figure 5.1 Block diagram of SHVC [109]

picture is obtained using the BL picture's interlayer processing (up-sampling) and the BL picture is reconstructed from BL's Decoded Picture Buffer (DPB). ILMP allows the EL to reuse the motion vectors of BL. The SHVC encoder uses the Advanced Motion Vector Prediction (AMVP) process during ILMP. The AMVP selects the four adjacent blocks to the current block and one block located at the same position in the frame at other times as candidate blocks. The additional coding information of BL, like inter-prediction direction, motion vectors, and reference index, can be used for coding the EL.

In SHVC, the quad-tree structure is used. Due to the BL and multiple EL layers, the RDO search process used in the SHVC provides more computational complexity than HEVC. In addition, the motion estimation further increases the encoding time.

The proposed method helps to reduce the complexity of SHVC by accelerating the ME, and CU size prediction process. The main contributions of this chapter are

1. We proposed the two stage Horizontal Subsampling Motion Estimation (HSME) that uses the Cross Diamond Unsymmetrical Octagonal (CDO) search pattern to obtain the motion vector with global minimum and to reduce the motion estimation time.

2. We establish a large database to train CNN and LSTM for predicting the CTU structure at inter mode which helps in reducing the complexity of SHVC.

3. We developed an Early terminated CNN+LSTM structure to predict the partitions

of CTU.

5.2 Proposed Work

In this section, the proposed method concentrates on reducing the motion estimation time and CTU structure prediction time. We propose the Horizontal Subsampling Motion Estimation (HSME) method to accelerate the motion estimation process. Then the Early Terminated CNN is explained, and finally, the ET-LSTM model is designed in combination with ET-CNN to predict the CTU partition structure in SHVC.

5.2.1 Horizontal Subsampling Motion Estimation (HSME)

In this method, two stage horizontal subsampling process is used which is shown in Fig. 5.2. The horizontal subsampling is the process of selecting only the even columns of the frame. Consider a frame (treated as Full Resolution (FR) frame) which is of size M×N. 'M' and 'N' represents the rows and columns of the frame. The frame undergoes the horizontal subsampling that results in the Half Resolution (HR) frame which is of size M×N/2. The resultant frame HR is converted to Quarter Resolution (QR) frame using the horizontal subsampling process. The QR frame is of size M×N/4. After creating the HR and QR frames, the motion estimation process is performed to determine the motion vector at each stage. The final goal of the two-stage horizontal subsampling process is to obtain the Best Motion Vector which is a global minimum. The searching operation is performed using the CDO search pattern (refer to section 4.3.1). The ME process starts at QR frame and ends at the FR frame. The ME process is explained below.

Step 1: The center biased search is used to determine the motion vector. Make the median predictor as a center and apply the CDO search pattern with the search range as 64 to the QR frame to obtain the motion vector. The motion vector obtained after the search process is treated as M1.

Step 2: Place the M1 as a center in the HR frame and perform the search process around M1 using the CDO to calculate the motion vector. The final motion vector calculated after search process is treated as M2. In step 2, the search range is 32.

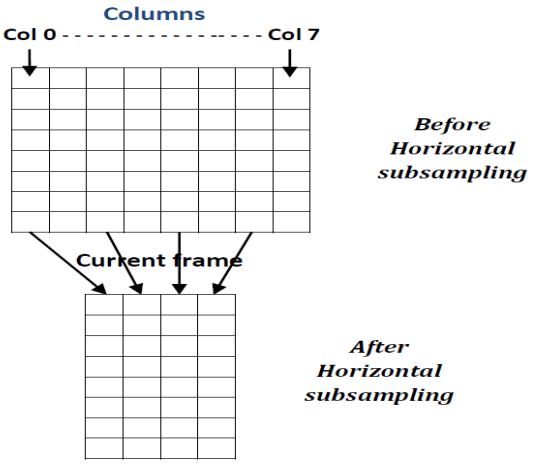


Figure 5.2 Horizontal subsampling

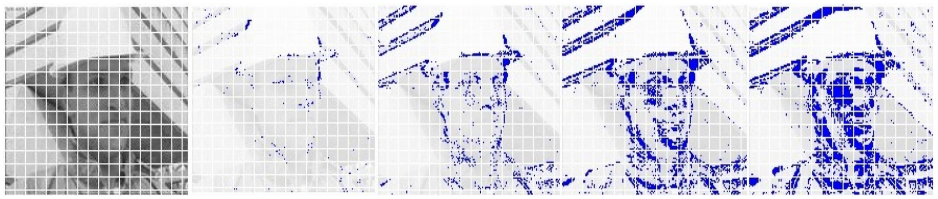
Step 3: Finally, place the M2 as a center in FR frame and determine the motion vector using the search pattern. The search range is 16 and the motion vector obtained after the motion estimation process is taken as the Best Motion Vector (BMV).

The advantage of this motion estimation process is the possibility of obtaining the global minima is remarkably high. The motion vector refinement at three frames leads to global minimum. The higher search range at QR frame and smaller search range at FR frame helps to reduce the motion estimation time.

5.2.2 Temporal correlation between frames

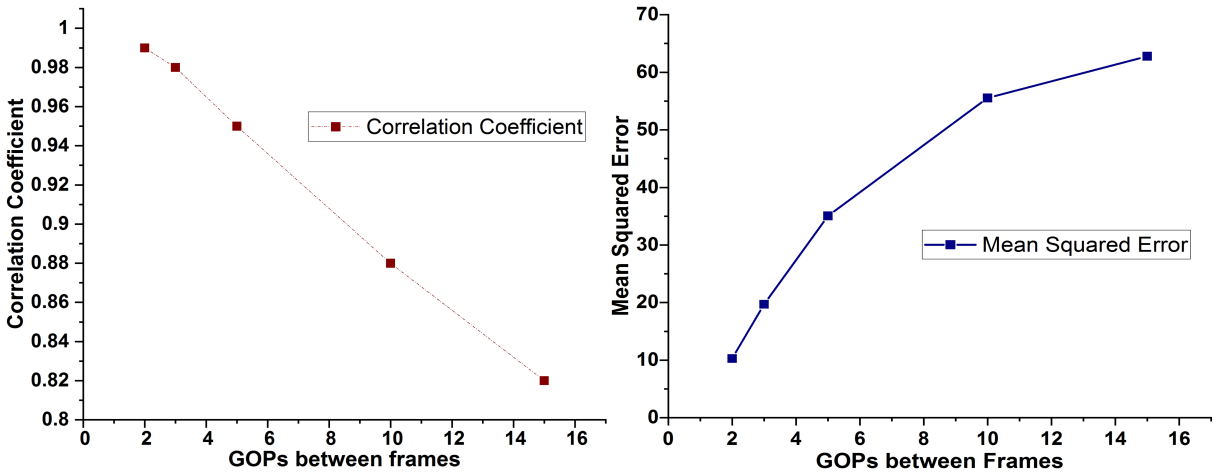
In the video sequences, the adjacent frames show similarity, and this similarity decreases as the distance between the frames increases. Fig. 5.3 shows the example CU partitions in adjacent frames. Consider 'frame 121' as a reference frame. The blue color indicates the dissimilarity between the frames. As shown in the figure, the dissimilarity increases as the temporal distance between the reference frame and the current frame increases.

We further evaluate the correlation between the two frames by considering 1 to 15 Group of Pictures (GOP). The evaluation is done in terms of mean square error (MSE) and correlation coefficient (CC). Fig. 5.4 shows that the CC is greater than zero, indicating a positive correlation between adjacent frames. The CC value decreases when the temporal distance increases. Moreover, the MSE value increases as the temporal dis-



(a) 121 frame **(b)** 122 frame **(c)** 124 frame **(d)** 127 frame **(e)** 130 frame

Figure 5.3 Example showing correlation between frames



(a) **(b)**

Figure 5.4 Comparison of (a) correlation coefficient vs distance between frames (b) Mean squared error vs distance between frames

tance increases.The CC and MSE values exhibit a long and short term dependency of CU partitions between neighboring frames.

5.2.3 ET-CNN structure

The Early Terminated CNN (ET-CNN) structure using the deep learning approach is shown in Fig. 5.5. The luminance of CTU (U) is given as an input to the ET- CNN structure. There are three branches in ET-CNN structure. This structure consists of preprocessing, convolution, concatenation, and fully connected layers. Each of the layers is discussed below.

1. **Preprocessing layer**

   Initially the CTU is pre-processed to reduce the variation of input samples. The

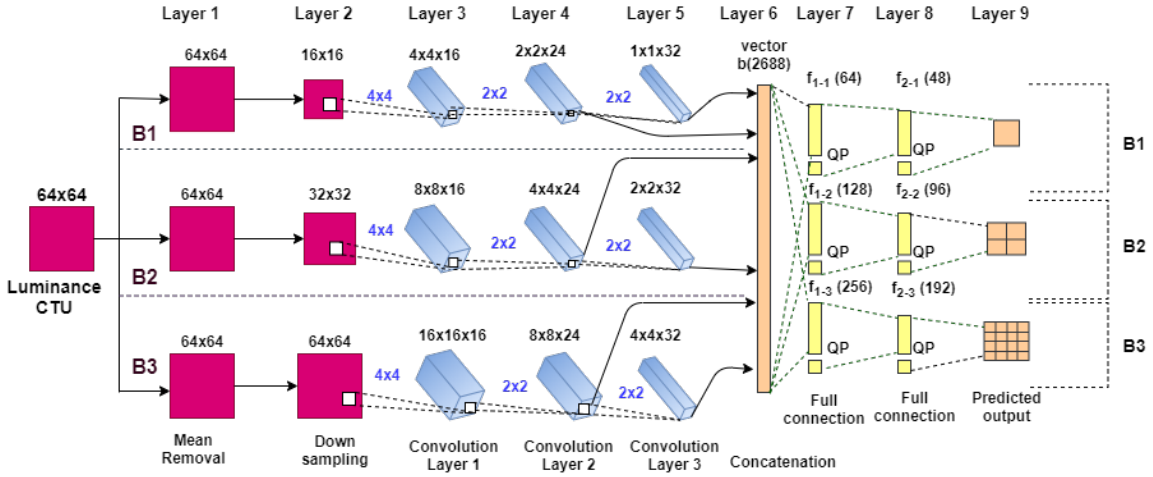


Figure 5.5 ET-CNN structure

pre-processing involves mean removal, and down-sampling process at layer 1 and layer 2 which is shown in Fig. 5.5. During the mean removal process, each CU is subtracted by their mean value to match the output structures at branches B1, B2 and B3 of layer 9. The down sampling is then applied to reduce the CU of size 64×64 to 32×32 and 16×16 at branches B2 and B1.

2. **Convolution layer**

   After pre-processing, the processed data is passed through three convolution layers. At layer 3, the 4×4 kernel is used for convolution with the data obtained from layer2 with 16 filters to extract low-level features. At each convolution layer, the same kernel is applied at three branches. At layers 4 and 5, the 2×2 kernel (with 24 filters at the fourth and 32 filters at the fifth layer) is used for convolution with feature maps to obtain high-level features. The width of the kernel is used as a stride length for non-overlapping operations.

3. **Concatenating layer**

   In this layer, the features obtained from three branches collected at layer four and layer 5 (convolution layers) are concatenated. The received feature maps together form as a single vector ‘b’. This vector is a combination of local and global features.

4. **Fully Connected Layers (FCLs)**

   The concatenating layer output vector ’b’ is given as an input to the FCL at three branches. There are two FCLs in ET-CNN. Based on the features of FCLs, the

ET-CNN predicts the CU structure, as shown in layer 9 of Fig. 5.5.

If the CU at B1 is not split, then the prediction operation at B2 and B3 can be skipped which saves the encoding time.

5.2.4 ET- LSTM design

Section 5.2.2 shows that the neighboring frames partition is correlated to each other. In this section, we propose the ET- LSTM approach that learns the dependencies of CU partitions between frames. The flowchart of the ET-LSTM approach is shown in Fig. 5.6. As shown in figure, the CTU residue is given as an input to the ET-CNN. To find the residue of CTU, we precode the current frame by forcing PU and CU sizes to 64×64. By precoding the frame, approximately 2 to 3 percent of total encoding time is increased, which is treated as an overhead. The parameters of the Early Terminated CNN are trained over residue using the database. The database is the collection of different video resolution sequences which are collected from Joint Collaborative Team on Video Coding (JCT-VC) test set and Xiph.org. All the sequences are encoded by using standard HM 16.5 at QPs of 22, 27, 32 and 37 respectively.

The ET-LSTM consists of three levels, which are represented by 'L.' Initially, the fully connected layer features $\left\{f_{1-L}\right\}_{L=1}^{3}$(shown in Fig. 5.5) is fed as an input to the LSTM cell in ET-LSTM. There are two fully connected layers after the LSTM cell. The FCL consists of QP value and frame order in the GOP. Note a one-hot vector indicates the order of the frame. The output of the LSTM cell and the first FCL is $f'_{1-L}t(b)$ and $f'_{2-L}t(c)$, respectively. The initial values of a,b, and c at level 1 are 64, 64, and 48. The $y^1(U,t)$ is the output of the second FCL, which represents either 1 or 0. The output predicted size of CU at each level when the $y^1(U,t)$ is zero is given below.

At L=1, the CU dimension is 64×64.

At L=2, the size of CU is 32×32.

At L=3, the CU size is 16×16.

At any level, if $y^1(U,t)$ is zero, stop the prediction operation and skip the remaining levels. For example, if $y^1(U,t)$ is zero at level L=1, skip the prediction operation for L=2 and 3 levels.This early termination helps in reducing the encoding time. If $y^1(U,t)$ is 1,

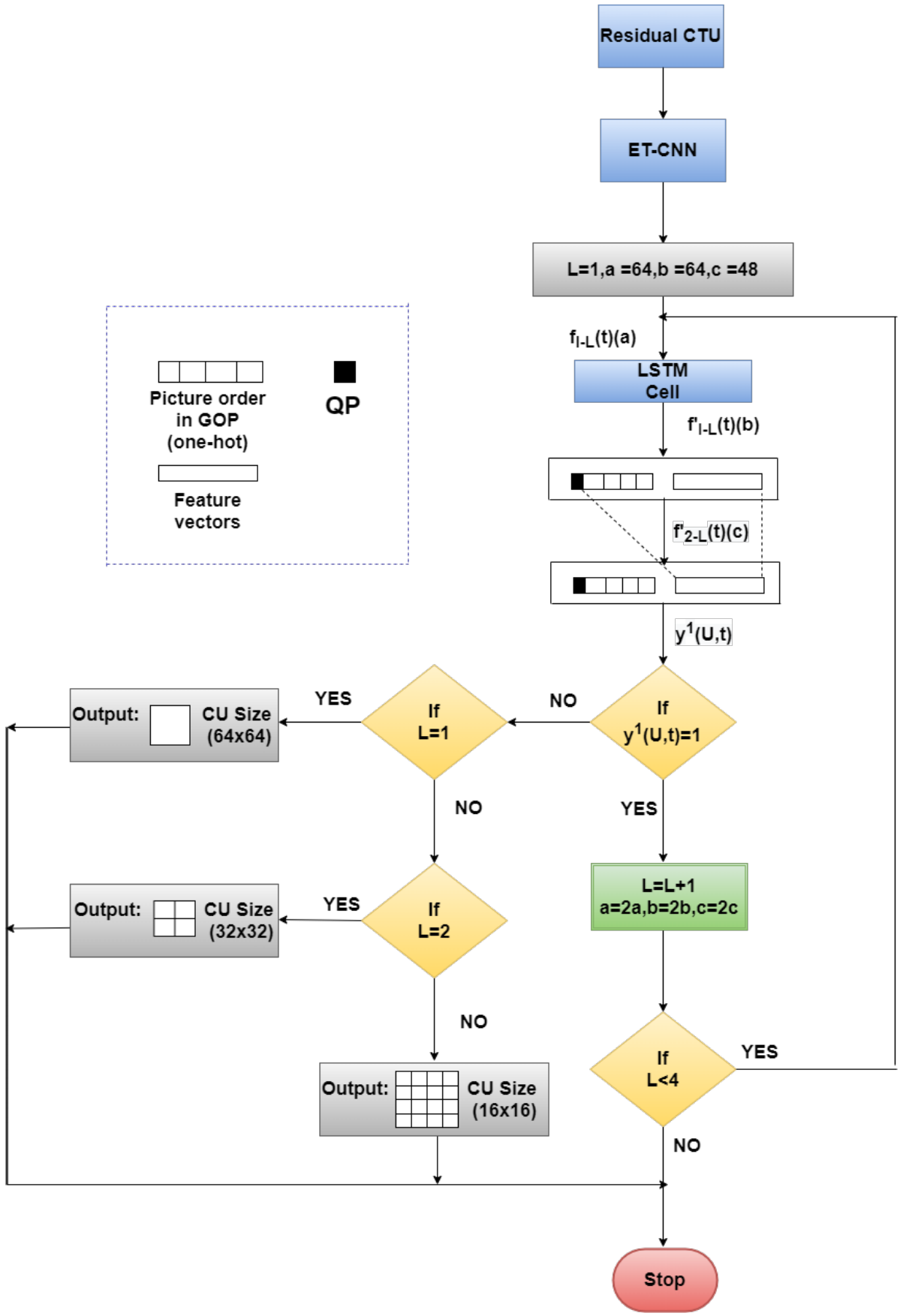


Figure 5.6 Flowchart of ET-LSTM structure

increment the level 'L' and update the values of a, b, and c to 2a, 2b, and 2c. If the L value is less than 4, repeat the operation with updated values by going back to the LSTM cell else stop the prediction operation. During the CU size prediction process, ET-LSTM considers the patterns of the co-located CTU present in the previous frame. The long and short term correlations of CU depth are obtained using the LSTM cells across distinct levels. The LSTM cell consists of three gates, namely the input gate $i_L(t)$, the forget gate $g_L(t)$, and the output gate $O_L(t)$. The three gates are obtained by using equations (5.1), (5.2) and (5.3).

$$i_L(t) = \sigma(W_i \cdot [f_{1-L}(t), f'_{1-L}(t-1)] + b_i) \tag{5.1}$$

$$O_L(t) = \sigma(W_o \cdot [f_{1-L}(t), f'_{1-L}(t-1)] + b_o) \tag{5.2}$$

$$g_L(t) = \sigma(W_f \cdot [f_{1-L}(t), f'_{1-L}(t-1)] + b_f) \tag{5.3}$$

Where, $\sigma(\cdot) \rightarrow$ sigmoid function, $W_i, W_o, W_f \rightarrow$ Three gates trainable weight parameters and $b_i, b_o, b_f \rightarrow$ Biases.

At frame t, the LSTM cell updates its state by using the three gates as

$$c_L(t) = i_L(t) \odot tanh(W_c \odot [f_{1-L}(t), f'_{1-L}(t-1)] + b_c) + g_L(t) \odot c_L(t-1) \tag{5.4}$$

Where, $W_c, b_c \rightarrow$ Parameters and biases of $c_L(t)$, $\odot \rightarrow$ Element-wise multiplication. $f_{1-L}(t)$, $f'_{1-L}(t-1) \rightarrow$ ET-CNN feature, LSTM cell feature output of last frame

The LSTM cell output $f'_{1-L}(t)$ is calculated by using equation (5.5).

$$f'_{1-L}(t) = O_L(t) \odot c_L(t) \tag{5.5}$$

To train the parameters present in equations (5.1),(5.2), (5.3), (5.4) and (5.5), ET-LSTM need 7,57,118, and 7,59,273 additions and multiplications. The cross-entropy is used as a loss function to train the parameters. The LSTM cell at each level is trained by optimizing the loss as

$$L = \frac{1}{RT} \sum_{r=1}^{R} \sum_{t=1}^{T} L_r(t) \tag{5.6}$$

The parameters are trained by considering 'R' training samples and 'T' frames. Finally, the ET- LSTM able to predict the CU size by using the trained LSTM cells.

Table 5.1 Experimental Conditions

Configuration	encoder_lowdelay_P_scalable
Codec version	SHM-12.1
QP	22, 27, 32, 37
CU Size (Max)	64×64
CU depth (Max)	4
Search range and GOP Size	64 and 8

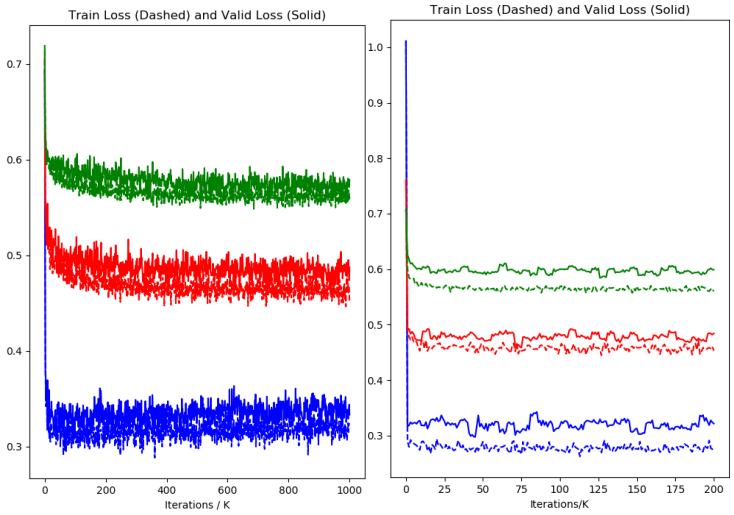
5.3 Experimental Results

The experimental results section presents the simulation results to evaluate the performance of the proposed method at intermode. The proposed method is compared with the state-of-the-art methods: Fast mode decision method by R. Bailleul [110] and Fast enhancement layer prediction by Chih-Hsuan Yeh [62].

Configuration of experiment: The proposed method is simulated in scalable HEVC software SHM-12.1 [111] to evaluate the performance using encoding time saving, bitrate, and PSNR parameters. The experimental parameters considered are shown in Table 5.1, and the 64-bit Windows-7 operating system with Intel(R) Core(TM) i5 is used for our experiments. The performance evaluation is done at four different Quantization parameter (QP) values i.e. at QP= {22, 27, 32, 37}. The proposed method is tested by using eighteen different video sequences which belong to five different classes. Class A belongs to (2560×1600) video resolution. Similarly, Class B, Class C, Class D and Class E belong to (1920×1080), (1280×720), (832×480) and (416×240) video resolutions, respectively.

These video sequences are compressed with encoder_lowdelay_P_scalable (LDP) and encoder_randomaccess_scalable10 (RA) configuration [112]. The efficiency of our proposed method is tested by using Bjontegaard's metric, which allows us to compute the average saving in bitrate [BD-BR] and average PSNR gain [BD-PSNR]. The equations (5.7), (5.8) and (5.9) represents the change in bitrate (Δ BR), saving in encoding time (TS) and change in YPSNR (Δ YPSNR).

$$\Delta\ BR(\%) = \frac{BR_{orig} - BR_{prop}}{BR_{orig}} \times 100 \tag{5.7}$$



(a) **(b)**

Figure 5.7 Training and validation loss of (a) ET-CNN (b) ET-LSTM

$$TS(\%) = \frac{T_{orig} - T_{prop}}{T_{orig}} \times 100 \tag{5.8}$$

$$\Delta\, YPSNR(dB) = YPSNR_{orig} - YPSNR_{prop} \tag{5.9}$$

Database for Inter-mode: The database contains 397 video files. Out of which, 300 video files of (112x112) size are taken from Stanford university, 18 sequences from the Joint Collaborative Team on Video Coding (JCT-VC) standard test set, and 79 video sequences of different resolutions from Xiph.org. The video sequences of the database belongs to different video resolutions: SIF (352×240), CIF (352×288) , 240p(416×240), 480p (832×480), 720p (1280×720), 1080p and WQXGA (2560×1600). The above sequences are randomly divided into validation (42 sequences), testing (30 sequences) and training (325 sequences). The above sequences are encoded at four QPs by HEVC to generate the CU depth data. Besides, 19,607,566 samples were collected for the LDP configuration.

Training Settings: In this paper, the ET-CNN is trained by utilizing the database (refer to section 5.3) at inter-mode. During the ET-CNN training, the validation datasets are tuned by the hyperparameters. For training the ET-CNN and ET-LSTM, the gradient descent algorithm's momentum is set to 0.8 and 0.9, respectively. Furthermore, the batch size is 32, and the learning rate is set to 0.01 for training the CNN and LSTM. Moreover, ET-CNN and ETLSTM is trained over 10,00,000 and 2,00,000 iterations, respectively.

Table 5.2 Experimental results of the HSME method for the JCT-VC video sequences under LDP configuration

Class	Video Sequence	Size	BD-BR (%)	BD-PSNR (dB)	TS(%)				
					QP=22	QP=27	QP=32	QP=37	TOT
A	PeopleOnStreet	2560x1600	4.12	-0.16	53.17	54.72	59.17	62.72	57.44
	Traffic		6.92	-0.24	46.52	49.76	54.29	58.86	52.35
B	Kimono	1920x1080	4.22	-0.49	36.71	42.46	49.32	51.33	44.95
	ParkScene		6.49	-0.27	24.32	29.19	37.82	44.17	33.87
	Cactus		6.58	-0.21	41.39	48.47	61.27	68.56	54.92
	BasketballDrive		8.41	-0.17	21.34	30.82	34.79	43.62	32.64
	BQTerrace		7.18	-0.24	24.86	40.83	58.74	68.00	48.10
C	BQMall	832x480	8.72	-0.39	36.41	48.76	58.74	65.91	52.45
	BasketballDrill		8.46	-0.21	32.00	59.86	65.31	74.39	57.89
	RaceHorses		7.95	-0.35	21.82	33.42	34.79	40.51	32.63
	PartyScene		5.17	-0.37	60.54	63.51	64.17	66.00	63.55
D	BasketballPass	416x240	6.34	-0.27	41.09	52.68	59.36	71.39	56.13
	BlowingBubbles		8.23	-0.25	31.03	34.39	49.56	60.24	43.80
	RaceHorses		8.17	-0.42	24.51	26.89	33.42	46.51	32.83
	BQSquare		7.41	-0.37	33.00	49.35	66.21	78.89	56.86
E	FourPeople	1280x720	3.26	-0.25	64.32	78.53	84.91	87.69	78.86
	KristenAndSara		3.71	-0.21	70.31	78.54	83.17	86.82	79.71
	Johnny		4.83	-0.17	60.91	74.68	82.53	84.31	75.60
Average			**6.45**	**-0.28**	**40.23**	**49.82**	**57.64**	**64.44**	**53.03**

Test Settings: CNN delivers the probability of binary output at each level when CTU is given as an input. The maximum and lower limits at level 1 are 0.6 and 0.4, respectively. Similarly, the upper and lower threshold for level 2 is 0.7 and 0.3, and for level 3, they are 0.8 and 0.2. The CU is split at each level based on the threshold value.

Evaluation on training performance and prediction accuracy: The training and validation loss for ET-CNN and ET-LSTM alongside the iterations are shown in Fig. 5.7. The training loss is calculated using equation (5.6) at each iteration. The figure shows that the loss converges after 3×10^4 iteration. The average accuracy of 88%, 83% and 78% obtained for CU partitions at levels L={1,2,3} while training the ET-LSTM.

Analysis of experimental results: Table 5.2 tabulate the simulation results of HSME method. The results show that the encoding time is saved by an average of 40.23%,

Table 5.3 Experimental results of the HSME method for the JCT-VC video sequences under RA configuration

Class	Video Sequence	Size	BD-BR (%)	BD-PSNR (dB)	TS(%)				
					QP=22	QP=27	QP=32	QP=37	TOT
A	PeopleOnStreet	2560x1600	6.18	-0.14	44.18	64.36	68.29	73.71	62.64
	Traffic		6.41	-0.41	53.98	57.60	62.74	65.58	59.98
B	Kimono	1920×1080	5.46	-0.57	41.37	53.70	44.15	60.31	49.88
	Parkscene		7.17	-0.24	25.63	35.71	46.34	51.17	39.71
	Cactus		7.45	-0.19	40.89	54.64	67.99	74.17	59.42
	BasketballDrive		8.91	-0.20	29.86	41.17	42.96	54.12	42.02
	BQTerrace		7.67	-0.23	36.82	51.86	68.21	76.05	58.24
C	BQMall	832×480	8.26	-0.29	39.55	50.41	64.93	71.85	56.69
	BasketballDrill		9.71	-0.32	33.16	39.63	55.43	65.82	48.51
	RaceHorses		8.15	-0.36	30.55	33.35	38.52	47.91	37.58
	PartyScene		3.46	-0.18	37.51	47.67	56.43	64.33	51.48
D	BasketballPass	416×240	6.30	-0.37	40.81	56.31	67.91	75.84	60.21
	BlowingBubbles		8.51	-0.41	35.20	44.30	48.96	58.74	46.80
	RaceHorses		7.85	-0.41	35.39	41.27	49.53	54.93	45.28
	BQSquare		7.21	-0.37	36.84	54.31	69.21	78.39	59.69
E	FourPeople	1280×720	4.09	-0.14	68.87	80.81	87.28	87.96	81.23
	KristenAndSara		4.31	-0.11	66.31	79.61	85.41	88.12	79.86
	Johnny		5.21	-0.21	66.81	79.65	84.93	87.64	79.75
Average			**6.80**	**-0.29**	**42.43**	**53.68**	**61.62**	**68.70**	**56.61**

49.82%, 57.64% and 64.44% at QP values of 22, 27, 32 and 37, respectively. The results also exhibit the domination of HSME method over the SHM-12.1 with more than 75% of TS for 1280×720 video sequences. The encoding time of SHM-12.1 is considerable decreased by an average of 53.03% using HSME method with 6.45% increase in BD-BR and 0.28dB loss in BD-PSNR for LDP configuration. Similarly, 56.61% of saving in encoding time with 6.80% rise in BD-BR and 0.29 dB loss in quality is observed for RA configuration in Table 5.3.

Table 5.4 and Table 5.5 shows the experimental results of the proposed approach in contrast to the conventional SHM-12.1. The results exhibit that the proposed method achieves a significant amount of saving in encoding time at each QP value with a small increase in bitrate and negligible decrease in YPSNR @ 2.25% and 0.02 dB for LDP

Table 5.4 Experimental results of the proposed method for the JCT-VC video sequences under LDP configuration

Class	Video Sequence	Size	Δ YPSNR (dB)	Δ BR (%)	TS (%)			
					QP=22	QP=27	QP=32	QP=37
A	PeopleOnStreet	2560x1600	-0.03	2.13	44.12	49.21	54.00	61.57
	Traffic		-0.02	3.41	39.45	45.35	52.61	60.43
B	Kimono	1920x1080	-0.04	0.55	39.68	41.39	45.95	55.11
	ParkScene		-0.03	1.43	49.00	49.09	40.75	63.80
	Cactus		-0.02	2.43	47.49	47.00	47.18	55.05
	BasketballDrive		-0.03	3.08	50.13	52.57	58.10	63.96
	BQTerrace		-0.02	4.85	34.82	55.76	42.14	55.70
C	BQMall	832x480	-0.01	1.78	28.98	39.64	36.83	51.22
	BasketballDrill		-0.05	2.10	42.89	45.81	49.81	51.82
	RaceHorses		-0.02	0.74	54.53	52.62	55.35	55.77
	PartyScene		-0.01	0.99	52.52	54.31	54.89	53.57
D	BasketballPass	416x240	-0.04	1.00	66.32	68.14	68.99	72.32
	BlowingBubbles		-0.03	0.56	62.62	54.99	54.84	55.21
	RaceHorses		-0.05	0.56	71.58	64.81	64.08	65.87
	BQSquare		-0.04	2.52	55.01	53.59	54.03	50.16
E	FourPeople	1280x720	-0.01	4.12	39.26	48.48	56.83	62.64
	KristenAndSara		-0.02	3.95	38.89	57.53	63.19	66.70
	Johnny		-0.03	4.34	47.49	61.83	66.04	72.11
Average			**-0.02**	**2.25**	**48.04**	**52.34**	**53.64**	**59.61**

configuration and 1.09% and 0.05 dB for RA configuration, respectively. The average time saving TS at QP={22,27,32,37} are 48.04%, 52.34%, 53.64% and 59.61%, respectively for LDP configuration. Similarly, 51.14%, 58.27%, 61.08% and 65.95% of saving in encoding time is observed for RA configuration.

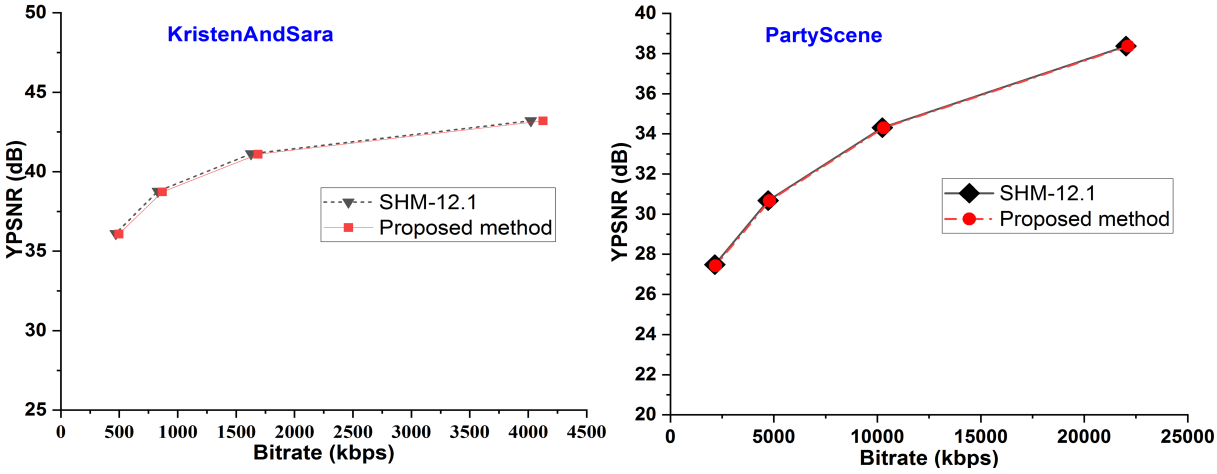
The proposed method is compared to state-of-the-art methods in [110] and [62] in Table 5.6 and Table 5.7. Our method outperforms the [110] and [62] algorithms in encoding time-saving TS for both LDP and RA configurations. For the proposed method under LDP configuration, the average time saving is 53%, which is high compared to the state-of-the-art methods. The average time savings of [110]and [62] algorithms are 44% and 38%, respectively. The positive values of BD-BR and negative values of BD-PSNR

Table 5.5 Experimental results of the proposed method for the JCT-VC video sequences under RA configuration

Class	Video Sequence	Size	ΔYPSNR (dB)	ΔBR (%)	TS(%)			
					QP=22	QP=27	QP=32	QP=37
A	PeopleOnStreet	2560x1600	-0.07	0.29	51.32	54.65	55.21	58.30
	Traffic		-0.08	1.33	48.05	48.31	49.90	56.24
B	Kimono	1920×1080	-0.02	0.38	43.22	54.12	60.09	67.04
	Parkscene		-0.06	1.35	34.43	56.44	60.32	66.01
	Cactus		-0.03	1.53	49.83	53.39	56.88	66.13
	BasketballDrive		-0.03	0.45	44.89	54.74	60.04	66.60
	BQTerrace		-0.06	2.52	56.87	56.96	57.52	61.65
C	BQMall	832×480	-0.04	0.54	33.12	43.80	52.44	62.86
	BasketballDrill		-0.04	1.31	43.49	46.32	51.61	55.81
	RaceHorses		-0.05	0.36	56.17	62.60	66.91	70.32
	PartyScene		-0.05	0.16	60.07	65.74	65.78	69.62
D	BasketballPass	416×240	-0.08	0.63	67.61	68.62	68.87	69.72
	BlowingBubbles		-0.04	0.18	58.68	59.52	59.61	63.86
	RaceHorses		-0.05	0.26	61.18	61.89	63.27	65.21
	BQSquare		-0.06	1.31	58.69	63.30	66.35	70.85
E	FourPeople	1280×720	-0.03	2.71	52.51	66.07	67.33	70.24
	KristenAndSara		-0.04	1.24	53.44	65.78	67.89	71.33
	Johnny		-0.04	2.98	47.13	66.68	69.40	75.36
Average			**-0.05**	**1.09**	**51.14**	**58.27**	**61.08**	**65.95**

represent the loss in coding performance. From Table 5.6, we notice that the average BD-BR value of our proposed method is 2.57%, which is very less compared to the BD-BR values of [110] and [62] methods @ 6.61% and 3.13%, respectively. In addition, our method incurs a loss of 0.14 dB BD-PSNR, which is slightly higher than [62] and better than the [110] method. Despite the decrease of BD-PSNR values for a few sequences compared to [110] and [62] shown in Table 5.6, the overall BD-PSNR and BD-BR are good.

The results in Table 5.7 show that the proposed method saves more encoding time compared to the previous methods for RA configuration. We can also observe that 1.68%



(a) (b)

Figure 5.8 Example RD curves of (a) KristenAndSara (b) PartyScene

Table 5.6 Comparison results of the proposed method and the state-of-the-art methods with SHM-12.1 as an anchor under LDP configuration

Class	Video Sequence	Size	Proposed			[110]			[62]		
			BD-BR (%)	BD-PSNR (dB)	TS (%)	BD-BR (%)	BD-PSNR (dB)	TS (%)	BD-BR (%)	BD-PSNR (dB)	TS (%)
A	PeopleOnStreet	2560x1600	**2.76**	-0.15	**52**	6.71	-0.19	44	2.97	**-0.11**	40
	Traffic		**3.27**	-0.19	**49**	5.94	-0.23	44	3.64	**-0.15**	41
B	Kimono	1920x1080	1.91	**-0.06**	**46**	4.54	-0.13	45	**1.18**	-0.09	35
	ParkScene		**2.17**	**-0.11**	**51**	7.77	-0.23	45	3.72	**-0.11**	42
	Cactus		**3.08**	-0.20	45	7.76	-0.14	**46**	3.27	**-0.06**	41
	BasketballDrive		1.35	-0.28	**56**	5.89	-0.09	44	**1.21**	**-0.05**	34
	BQTerrace		2.22	-0.41	**47**	5.81	**-0.12**	46	**0.85**	-0.03	33
C	BQMall	832x480	**2.21**	**-0.09**	39	6.68	-0.26	**44**	4.54	-0.18	39
	BasketballDrill		3.84	-0.14	**48**	8.23	-0.32	44	**1.31**	**-0.07**	33
	RaceHorses		**1.12**	**-0.04**	**55**	5.48	-0.25	43	2.32	-0.10	37
	PartyScene		**1.31**	**-0.06**	**54**	5.29	-0.27	44	3.32	-0.17	38
D	BasketballPass	416x240	**1.78**	**-0.09**	**69**	6.54	-0.34	42	4.54	-0.23	39
	BlowingBubbles		**1.17**	**-0.04**	**57**	6.85	-0.30	43	3.64	-0.16	36
	RaceHorses		**0.89**	**-0.04**	**67**	7.63	-0.44	41	3.92	-0.22	35
	BQSquare		3.01	-0.14	**53**	5.99	-0.26	43	**1.08**	**-0.09**	30
E	FourPeople	1280x720	4.87	-0.17	**52**	7.02	-0.17	45	**4.82**	**-0.12**	44
	KristenAndSara		**5.00**	-0.16	**57**	6.68	**-0.15**	44	7.58	-0.17	45
	Johnny		4.37	-0.15	**62**	8.19	-0.14	45	**2.49**	**-0.06**	39
Average			**2.57**	-0.14	**53**	6.61	-0.22	44	3.13	**-0.12**	38

rise in BD-BR and 0.10 dB drop in YPSNR which is less than the approach in [110] and slightly higher than the method in [62]. Our proposed method is decreasing more

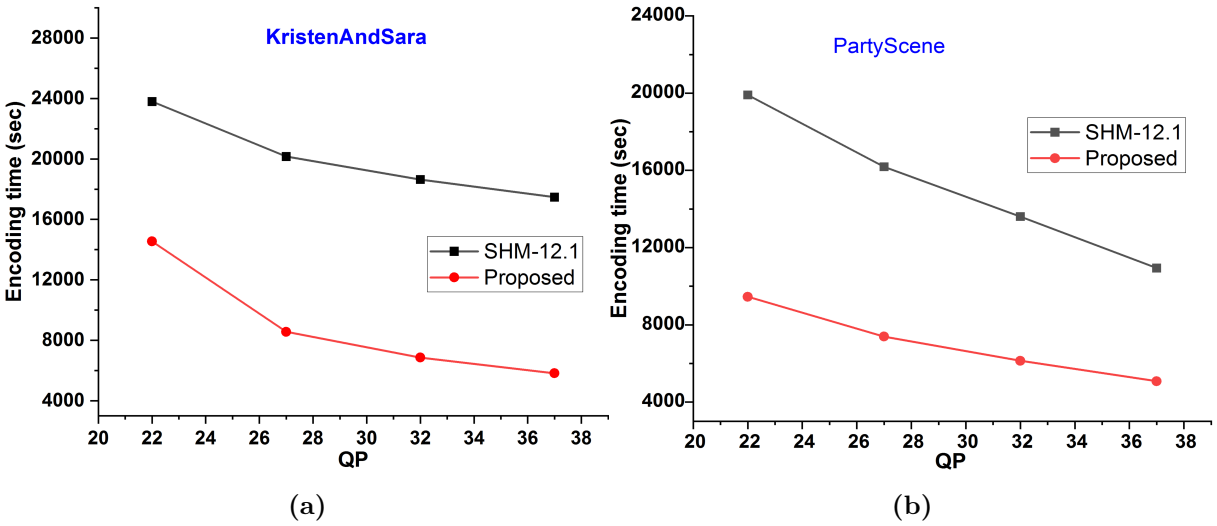


Figure 5.9 Example encoding time curves of (a) KristenAndSara (b) PartyScene

Table 5.7 Comparison results of the proposed method and the state-of-the-art methods with SHM-12.1 as an anchor under RA configuration

Class	Video Sequence	Size	Proposed			[110]			[62]		
			BD-BR (%)	BD-PSNR (dB)	TS (%)	BD-BR (%)	BD-PSNR (dB)	TS (%)	BD-BR (%)	BD-PSNR (dB)	TS (%)
A	PeopleOnStreet	2560x1600	2.42	-0.13	**55**	7.59	-0.29	43	**0.71**	**-0.03**	29
	Traffic		2.95	-0.19	**51**	7.48	-0.19	44	**0.91**	**-0.04**	37
B	Kimono	1920×1080	0.62	-0.04	**56**	4.68	-0.11	44	**0.84**	**-0.03**	33
	Parkscene		1.68	-0.14	**54**	6.36	-0.16	45	**0.91**	**-0.03**	35
	Cactus		2.79	-0.18	**57**	7.48	-0.15	46	**0.68**	**-0.02**	33
	BasketballDrive		0.74	**-0.05**	**57**	6.52	-0.12	43	**0.63**	-0.06	33
	BQTerrace		**1.17**	-0.19	**58**	6.13	-0.12	47	1.21	**-0.05**	35
C	BQMall	832×480	1.50	**-0.06**	**48**	7.18	-0.25	44	**1.34**	-0.07	31
	BasketballDrill		1.31	**-0.07**	**49**	7.86	-0.32	45	**1.23**	-0.08	34
	RaceHorses		1.72	-0.09	**64**	6.15	-0.26	44	**0.91**	**-0.03**	28
	PartyScene		1.13	**-0.05**	**65**	5.31	-0.24	48	**1.03**	**-0.05**	33
D	BasketballPass	416×240	1.36	-0.07	**69**	6.54	-0.35	42	**1.16**	**-0.06**	33
	BlowingBubbles		**1.05**	**-0.04**	**60**	6.16	-0.24	44	1.33	-0.05	31
	RaceHorses		**1.12**	**-0.05**	**63**	7.27	-0.41	**39**	1.26	-0.07	28
	BQSquare		2.44	-0.23	**65**	5.12	-0.18	45	**0.73**	**-0.03**	33
E	FourPeople	1280×720	1.60	-0.06	**64**	6.21	-0.15	44	**0.64**	**-0.02**	39
	KristenAndSara		1.44	**-0.04**	**65**	5.71	-0.14	45	**0.81**	-0.05	40
	Johnny		2.46	-0.05	**64**	6.31	-0.11	45	**1.12**	**-0.03**	41
Average			1.68	-0.10	**59**	6.44	-0.21	44	**0.97**	**-0.04**	34

complexity of SHVC compared to state-of-the-art methods for most of the sequences at different resolutions and QPs.

The proposed method is also evaluated using RD curves in addition to encoding time and quality. The example RD curves and encoding time comparison graphs for KristenAndSara and PartyScene video sequences are shown in Fig. 5.8 and Fig. 5.9. Fig. 5.8 (a) and (b) shows that the RD curve of the proposed method almost overlaps the SHM-12.1. This represents the proposed method can generate the same quality as the standard SHM-12.1. The improvement in RD performance is achieved due to the high prediction accuracy of CU partitions. Fig. 5.9 (a) and (b) show that the proposed method takes less time to encode than SHM-12.1 for KristenAndSara and PartyScene video sequences.

5.4 Conclusions

SHVC can provide high efficiency without much decrease in perceptual video quality. Despite the increase in efficiency, the complexity of SHVC also increases. In this chapter, we have proposed the HSME method that speed up the motion estimation process and obtains the motion vector with global minimum. In addition, we have designed an ET-LSTM network to predict the CU partition with less complexity. The ET-CNN learns the CU partition from residual CTU by taking the output features of a fully connected layer present in the ET-CNN. The proposed method which is the combination of HSME and ET-CNN+ET-LSTM has reduced the encoding time by 53% for LDP and 59% for RA configuration, which is high compared to state-of-the-art methods.

Chapter 6

Surgical Incision Region Encoding using Scalable High Efficiency Video Coding

6.1 Introduction

Surgical telementoring has acquired heaps of interest, particularly in rural areas. The problems that arise during the surgical procedures are complex for the inexperienced surgeon to handle. Telementoring is the process of transferring the knowledge from the experienced surgeon to the novice who is present at a distant location. However, the limited bandwidth resources present in the remote areas make the telementoring system difficult to implement. The efficient telementoring system requires transmission of Region of Interest (ROI) of video with high quality in a limited bandwidth. In this chapter, the surgical incision region is referred to as ROI. Approximately 5 Mbps bandwidth is required to transmit the high-quality videos for telementoring applications which is very difficult to achieve in disaster-affected areas.

The Scalable extension of HEVC (SHVC) [113, 114] can be used in such a case that provides highly scalable coding efficiency and allows the transmission of a single video with different resolutions in a single bitstream. However, the complexity of SHVC makes it unsuitable for real-time applications. The complexity is increased mainly due to the Rate-Distortion Optimization (RDO) search process.

The surgical video frame shown in Fig. 6.1 consists of the background region and the surgical incision region. The surgical telementoring system requires the transmission

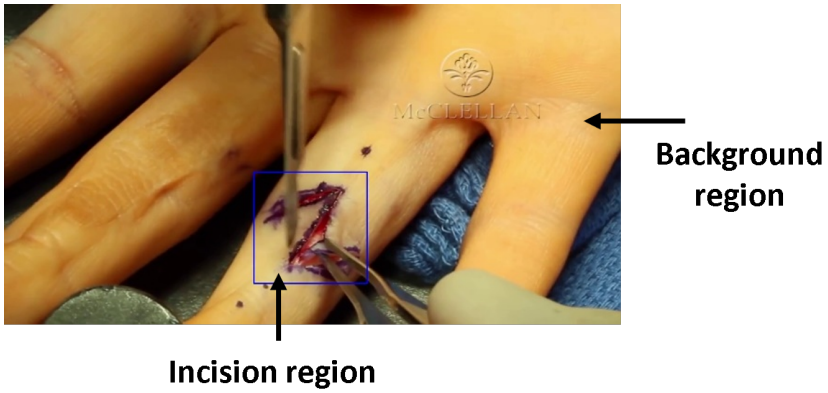


Figure 6.1 Surgical video frame with surgical incision and background region

of the surgical incision region with high quality. Several segmentation techniques can be used to identify the incision region in the video. The Mean Shift (MST) [115] algorithm is used in the medical image field to identify the surgical region. However, this technique is incapable of processing the large image of data, which results in the segmentation of the ROI with less accuracy. The Convolutional Neural Network (CNN) helps segment the ROI in computer vision applications; however, it increases the computational complexity.

The above techniques help to encode the ROI with high quality and the background region with low quality. As the background region does not contain any valuable information, the pixel values of the background region can be made zeros, resulting in the decrease of bitrate. The ROI region in the surgical videos moves slowly, and less abrupt changes can be observed throughout the video sequence. Hence, the object tracking technique can be used to track and extract the ROI in the surgical video.
The main contributions of this chapter are as follows

1. The Kernelized Correlation Filter (KCF) object tracking technique is used to track the ROI in the surgical video sequence.
2. A large database with 397 video sequences is created to train the CNN.
3. The Long- and Short-Term Memory (LSTM) network in combination with CNN is designed to reduce the complexity of SHVC.

The proposed method extracts the ROI from the surgical video frames using the KCF object tracker and encodes the ROI using SHVC with less complexity using the deep learning CNN+LSTM technique.

6.2 Background Work

The surgical telementoring system requires the surgical incision region to be encoded with high quality. Several authors suggested different algorithms to code the Region of Interest (ROI) in a video with high quality and the remaining region with low quality. The authors in [116] extracted the facial features using MST algorithm and encoded them with high quality. The background region is encoded in lower quality. In [117], the authors used the 3D morphological technique to segment the ROI region in colon computed tomography (CT). The researchers in [118] used the H.264 encoder to encode the segmented part of echocardiogram and CT video sequences. The segmentation is done using the image processing techniques like squared gradient, Sobel operators, and thresholding techniques. The Nearest Neighbor (NN) classifier is used in [119] to extract the ROI region in ultrasound videos. The results show that the bitrate is reduced by an average of 13.52% at the cost of high computational complexity.

In [120], a new method is proposed that allows the manual selection of the desired region in the video and encodes the selected area with high quality for surgical telementoring application. In [121], the authors designed a method that adaptively sets the ROI location and resolution based on the predefined settings. The desired ROI location is obtained by removing the background region, and then the inter-layer prediction operation is performed on the selected ROI region. This method saves the bitrate by 33.48%. The kernel-based MST method is used in [122] that requires the user interaction to select the desired ROI and the related resolution. The authors encode the selected ROI using the Huffman encoding technique. The authors in [123] use the non-parametric segmentation to detect the surgical incision region by considering the physiological behavior of the visual system and encodes the ROI with high quality. The authors in [124] developed a method for surgical telementoring application that performs image compression, image denoising, and image segmentation operations on computed tomography images for the diagnosis of congenital heart disease. The researchers in [125] reports the augmented reality system that uses the 3D tracking module and the Microsoft HoloLens for training and the telementoring surgery. The authors in [126] developed the deep CNN, which is SegNet that uses 26 convolutional layers for image segmentation. This method is computationally expensive.

The authors in [127] use the probability of the human attention over the frames to allocate coding bits using the visual saliency map scheme. The experimental findings indicate 43% saving in encoding time and 23% reduction in bitrate. In [128], the smartphones are used to capture the wound image. The wound part is segmented using the mean shift algorithm, and the red-yellow-black color model analyzes the wound. Similarly, the authors in [129] and [130] use the mean shift algorithm to detect the boundary of the foot injury and for the classification of skin tissue. In [131], the authors use the CNN approach to segment and analyze the wound region. The researchers in [132] proposed the CNN method that uses the convolutions for the extraction of multiple-level features for Diabetic Foot Ulcer (DFU) classification.

The CNN technique efficiently separates the surgical incision region from the background region. However, high computational complexity makes them less suitable for real-time applications. The background region doesn't contain vital information. The encoding of the background region increases the bit rate. Some of the authors use the HEVC encoder to encode the surgical videos with high quality. But the RDO search process in the HEVC increases the complexity that in turn increases the encoding time. We proposed the efficient surgical telementoring system that encodes the ROI with high quality in less time using SHVC with CNN+LSTM for real-time performance.

6.3 Proposed Method

In this section, we first analyze the correlation between the frames for surgical and general video sequences. Then, the object tracking using the KCF tracker is used to detect the surgical incision region. Finally, the SHVC using CNN and LSTM (refer to section 5.2.3 and 5.2.4) is used to encode the surgical incision region with less complexity. The framework of the proposed method is shown in Fig. 6.2. The operation of the proposed method is explained in the following subsections.

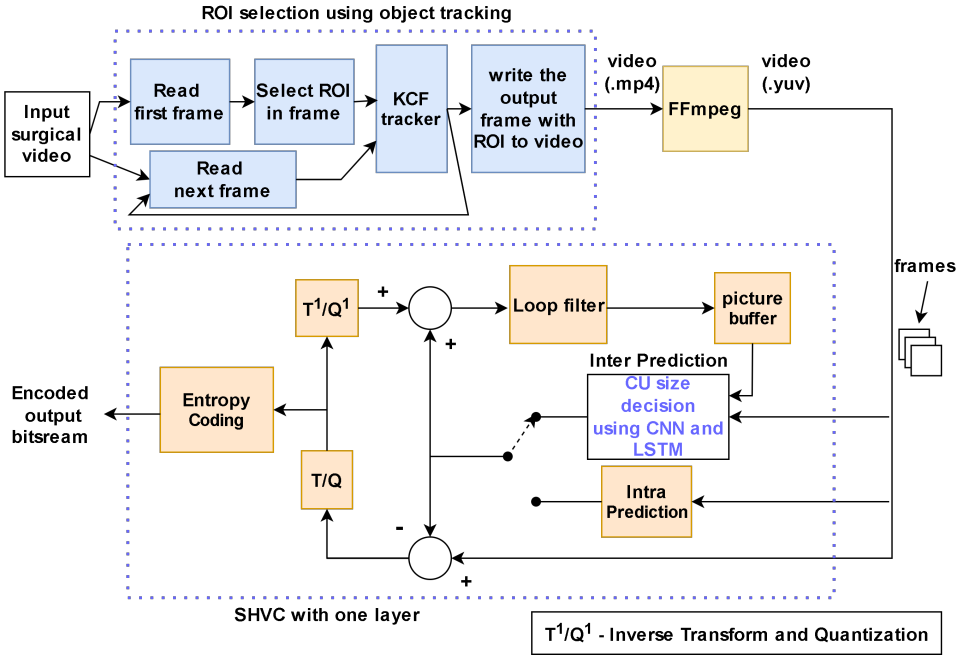


Figure 6.2 Framework of the proposed method for surgical telementoring application

(a) Frame 1 **(b)** Frame 40 **(c)** Frame 200

Figure 6.3 Example frames of the fast motion Basketball video sequence

6.3.1 Analysis of Correlation between Frames

This section analyzes the correlation between the frames for the surgical and general video sequences. The surgical video sequence "NuGrip Arthroplasty" and the general video sequence "BasketballPass" with frames at a different distance are shown in Fig. 6.4 and Fig. 6.3. The frames in the surgical video show that the surgical incision region movement is very small throughout the sequence.

The conventional object tracking technique is sufficient to track and extract the surgical region. However, the objects in the general video sequence move rapidly, which requires deep learning segmentation techniques to detect the boundary of ROI. The deep learning techniques segment the ROI more accurately at the cost of high computational complexity. Fig. 6.5(a) and Fig. 6.5(b) show the correlation coefficient and Mean Squared

(a) Frame 1 **(b)** Frame 40 **(c)** Frame 200

Figure 6.4 Example frames of the surgical NuGrip Arthroplasty video sequence

Error (MSE) curves of BasketballPass and NuGrip Arthroplasty video sequences. The figures show that the correlation coefficient is high for NuGrip Arthroplasty compared to the BasketballPass sequence. The high correlation coefficient is observed due to the small movement of ROI in a surgical video sequence with the background region remaining almost constant. The correlation coefficient and MSE are inversely proportional to each other. The NUGrip Arthoplasty produces less MSE due to the high similarity between the frames.

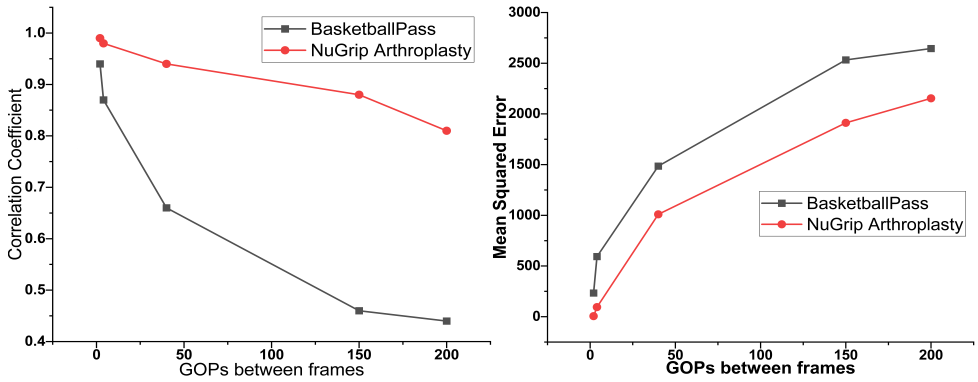


Figure 6.5 Correlation and Mean Squared Error curves of general and surgical video sequence at different distance between the frames. Note that the BasketballPass represents the general video sequence and NuGrip Arthroplasty represents the surgical video sequence

6.3.2 ROI Tracking using KCF Tracker

The analysis in section 6.3.1 shows that the object movement is very small, and the background region remains almost constant throughout the sequence. Hence, the object tracking techniques can be used to track the ROI effectively with less computational

complexity. We use the Kernelized Correlation Filter (KCF) tracker to track the ROI throughout the video sequence. The KCF tracker has the advantage of high efficiency.

The Flowchart of the object tracking algorithm using the KCF tracker shown in Fig. 6.6 is discussed in the following steps.

Step 1: Take the surgical video as an input.

Step 2: Read the first frame and select the ROI using the rectangular bounding box as shown in Fig. 6.1.

Step 3: Initialize the KCF tracker.

Step 4: Read the next frame of the surgical video sequence.

Step 5: Create the mask with the size of the surgical frame and make all the pixel values of the mask zero.

Step 6: Check whether the ROI is tracked by the KCF tracker using the bounding box coordinates. If tracked, go to step 7. Otherwise, treat the mask as output and go to step 9.

Step 7: Identify the tracked bounding box coordinates and change the pixel values in the bounding box coordinates of the mask to 255.

Step 8: Find the output by applying AND operation between mask and frame.

Step 9: Write the output to the video.

Step 10: If the frame count reaches the last frame of the surgical video sequence, Stop the operation. Otherwise, move to step 4.

KCF Tracker

We base our methodology on KCF [133], which displays amazingly real-time performance and accuracy comparative with the new top-performing trackers. The purpose of the correlation filter is to estimate an optimal filter to produce the desired response for the image input. The desired response is of Gaussian shape at the ROI location. The samples for training are obtained by cyclically shifting the whole area around the object. During

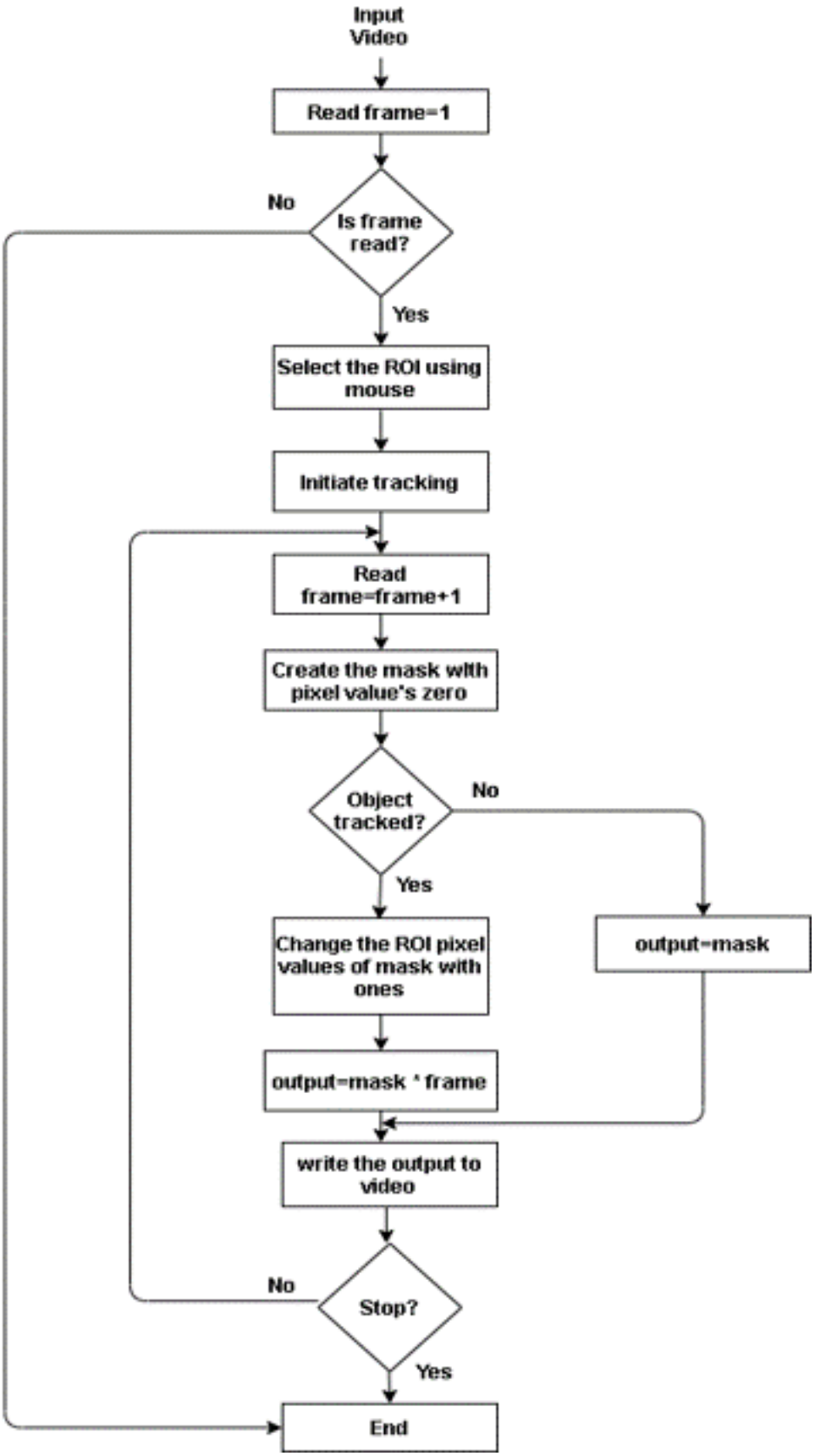


Figure 6.6 Flowchart of ROI tracking using the KCF tracker

testing, the position where the maximum filter response is obtained represents the target location. The KCF tracker has the advantage of high computational efficiency, obtained by utilizing a Discrete Fourier Transform (DFT). The "kernel trick" is also deployed to improve the performance of the KCF tracker further. The KCF tracker is summarized below.

Consider the cyclic shift matrix X with the dimension of M×N. 'M' and 'N' represent the total number of rows and columns of the matrix. Each row represents the one-dimensional data. Let the data in the first row is x=[x_1, x_2, x_3,....,x_{n-1}, x_n] and the data in the remaining rows represents the cyclic shifted data of previous row. The cyclic shifted data of the first row is [x_n, x_1, x_2, x_3,.....,x_{n-1}]. All the cyclic shifted rows together form a cyclic shift matrix.

During training, the tracker learns an optimal filter 'w' that can be found by minimizing the regression error as

$$\min_{w} \sum_{j} (w\psi(x_j) - y_j)^2 + \lambda \|w\|^2 \tag{6.1}$$

Where, $\psi(x_j)$ is training samples, y_j is regression labels and $\lambda \geq 0$ represents the regularization parameter.

As the circulant matrix can be diagonalized with the help of a Discrete Fourier Transfer (DFT) matrix, 'w' in equation (6.1) can be calculated quickly using the Fourier domain operation as

$$\hat{w} = \frac{\hat{x} \odot \hat{y}}{\hat{x} \odot \hat{x}^* + \lambda} \tag{6.2}$$

Where, $\odot \rightarrow$ Element-wise product, * and $\hat{}$ indicates conjugate and DFT operation. In KCF tracker, the 'Kernel trick' is applied to improve the performance of the filter in the non-linear regression. Now the 'w' becomes

$$w = \sum_{j} \alpha_j \psi(x_j) \tag{6.3}$$

Where, α = Dual parameter of 'w'. For the circulant matrix, the solution of the regression $\hat{\alpha}$ can be obtained as shown in equation (6.4).

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda} \tag{6.4}$$

Here, k^{xx} is the first row of the kernel matrix, $\hat{}$ represents the DFT operation.

After training, the detection operation is applied on the image patch 'z' in the upcoming frame within a M×N window. Then the response is obtained as :

$$f(z) = DFT^{-1}(\hat{k}^{xz} \odot \hat{\alpha}) \tag{6.5}$$

Where, $\hat{k}^{xz}$ is kernel correlation. Hence, the location of the target can be determined in each frame based on the maximum response ($f(z)_{max}$). Finally, to maintain the appearance of the target, the linear interpolation is used to update the sample template $\hat{x}$ and the dual coefficients $\hat{\alpha}$ with η as a fixed learning rate is given in equation (6.6) and (6.7)

$$\hat{x_t} = \hat{x}_{t-1}(1-\eta) + \eta\hat{x}_t \tag{6.6}$$

$$\hat{\alpha_t} = \hat{\alpha}_{t-1}(1-\eta) + \eta\hat{\alpha}_t \tag{6.7}$$

The KCF tracker works efficiently when the surgical video sequence contains slow-moving objects and a smaller number of scale changes.

6.3.3 FFmpeg

After object tracking, FFmpeg helps to convert the output video to the YUV video sequence. FFmpeg is the leading multimedia framework, able to encode, decode and transcode videos with different formats. FFmpeg can be obtained from the website https://www.ffmpeg.org/. In FFmpeg, the video sequence from 'mp4' format to 'yuv' format can be converted by using the command below

ffmpeg -i input.mp4 -c:v rawvideo -pixel_format yuv420p output.yuv

The output YUV sequence is encoded by SHVC using ET-CNN and ET-LSTM (refer to section 5.2.3 and 5.2.4).

Table 6.1 Experimental Conditions to simulate the proposed method in SHM-12.1

Configuration	encoder_lowdelay_P_scalable
Codec version	SHM-12.1
Number of layers in SHVC	2
QP	22, 27, 32, 37
CU Size (Max)	64×64
CU depth (Max)	4
Search range and GOP Size	64 and 8

6.4 Experimental Results

This section presents the experimental findings to analyze the performance of the proposed method. The surgical telementoring system requires the wireless transmission of high-quality video with less bit rate. However, it is extremely difficult to encode the entire frame with high quality and less bit rate. We use the KCF tracker to track the ROI in the frames of the video, extract it and encode it with SHVC by maintaining high quality and less bitrate.

Configuration of Experiment: The scalable HEVC reference software SHM-12.1 [111] is used to simulate the proposed method. The experiment is performed on Intel Core i7 CPU using experimental parameters shown in Table 6.1. The proposed method is analyzed in terms of Bit Rate (BR) saving and change in Peak signal-to-noise ratio (ΔPSNR), which can be calculated using equations (6.8) and (6.9).

$$BR\,Saving(\%) = \frac{BR_{orig} - BR_{prop}}{BR_{orig}} \times 100 \tag{6.8}$$

$$\Delta\ PSNR(dB) = PSNR_{orig} - PSNR_{prop} \tag{6.9}$$

The PSNR and BR can be measured using equations (6.10) and (6.11).

$$PSNR = 10log_{10}\frac{(2^{bitdepth} - 1)^2 \times W \times H}{\sum_i (O_i - D_i)^2} \tag{6.10}$$

Table 6.2 Experimental results of Default SHM 12.1 and proposed method for surgical video sequences

Video	SFF				PFROI				PROI			
	Size	BR	PSNR	Enc time	Size	BR	PSNR	Enc time	Size	BR	PSNR	Enc time
Z-Plasty [134]	1280x720	5335.61	47.46	33480.15	1280x720	1489.99	56.99	8015.82	280x254	1333.52	48.22	1694.59
Digital nerve [135]		6754.70	46.25	34496.92		481.04	62.22	4933.73	128x116	410.42	46.96	343.85
Flexor [136]		11257.54	44.73	40585.82		3493.40	51.70	13786.04	410x458	3121.35	45.08	7257.96
Finger [137]		7692.73	45.60	38648.90		703.30	60.33	5483.99	148x118	547.90	45.78	425.39
Flexor Tendom [138]		3321.21	47.83	31556.20		1345.57	56.51	7641.21	416x196	1131.67	48.53	1498.82
Volar wrist [139]		48983.94	45.39	67907.25		6736.62	56.74	13099.15	410x300	782.22	48.05	1642.18
Arthroplasty [140]		5291.82	48.30	31542.49		997.25	59.01	6276.50	264x154	907.21	48.90	1017.26
Tendon saw injury [141]		74974.65	44.61	77555.37		21481.50	51.32	24137.70	526x574	3370.66	47.09	7495.33
Subcuticular [142]		121738.53	43.04	77670.56		19603.82	52.03	22111.30	454x456	3810.55	45.74	5425.51

Where, bitdepth → each pixel bit depth, W → Width, H → Height, O_i → reference frame pixel value, D_i → Decoded frame pixel value, i → pixel address.

$$BitRate(BR) = \frac{W \times H \times T_F}{bpp \times fps \times 1000}(kbps) \tag{6.11}$$

Where, bpp → bits per pixel, fps → frames per second, W → Frame width, H → Frame height, T_F → Total number of frames. Bitrate is measured in 'kbps.'

In addition, the average saving in bitrate (BD-BR) and average PSNR gain (BD-PSNR) quantifies the RD performance loss.

Analysis of Experimental Results:

Table 6.2 shows the experimental results of the full-frame coding using SHVC (SFF), proposed Frame with only ROI coding using SHVC (PFROI), and proposed ROI coding using SHVC (PROI). The PFROI coding is done by selecting and tracking the ROI using the KCF tracker, making the pixels other than ROI zero, and coding using SHVC. The PROI coding is performed by tracking the ROI from the surgical video using the KCF tracker, cropping the ROI, and coding the ROI using SHVC. Fig. 6.7 shows the Full frame, mask, Frame with ROI, and cropped ROI.

During the simulation process, only one layer is chosen in SHVC, which acts as a

Table 6.3 Comparison of proposed method and state-of-the-art methods segmentation accuracy for surgical videos

Video	Prop	SegNet	S-CNN	MST	Prop	SegNet	S-CNN	MST
	Pixel Accuracy				fIOU			
Z-Plasty	97.40	96.7	98.6	94.3	98.21	98.1	97.5	92.0
Digital Nerve	98.21	97.4	98.3	96.4	97.48	97.6	97.3	94.4
Flexor	97.85	96.9	97.1	93.0	96.78	95.4	94.4	86.3
Finger	97.18	98.0	98.2	94.4	98.55	97.7	98.3	94.3
Flexor Tendom	96.42	96.3	98.1	86.9	96.80	94.3	96.5	84.1
Volar Wrist	96.68	96.8	97.4	93.2	97.27	95.4	96.3	91.2
Arthroplasty	97.97	97.3	98.6	95.1	96.23	97.4	97.0	93.6
Tendon Saw Injury	98.57	97.3	98.4	94.6	96.96	97.0	97.5	92.4
Subcuticular	96.85	98.0	97.3	95.1	97.93	98.3	96.9	92.9
Average	**97.45**	**97.18**	**98.0**	**93.66**	**97.35**	**96.80**	**96.85**	**91.14**

(a) **(b)** **(c)** **(d)** **(e)**

Figure 6.7 ROI extraction process using object tracking involves (a) Original frame (b) ROI selction in original frame (c) Mask (d) Output frame with tracked ROI (e) Output ROI cropped frame

single layer HEVC to encode surgical videos. Based on the analysis present in [143], we choose QP=20 for simplicity to encode surgical videos with high quality.

Table 6.3 presents the segmented accuracy results of the proposed method, SegNet [126], S-CNN [143], and the MST [115] techniques. The segmented accuracy is calculated using pixel accuracy, and frequency weighted IoU (fIoU). The pixel accuracy and fIoU can be measured using equations (6.12) and (6.13).

$$Pixel\ accuracy = \frac{TN + TP}{TP + TN + FP + FN} \tag{6.12}$$

Here, TP → True Positive, TN → True Negative, FP→ False Positive and FN →

False Negative.

$$fIoU = (\sum_{k} T_k)^{-1}(\frac{\sum_n T_n p_{nn}}{T_n + \sum_m p_{nm} - p_{nn}}) \tag{6.13}$$

Where, p_{nn}=number of correctly identified pixels, p_{nm}= number of pixels rejected incorrectly for class m and T_n = Total number of pixels in class n.

From Table 6.3, the results show that the proposed approach achieved higher pixel accuracy compared to the SegNet and MST techniques. Even though the pixel accuracy is slightly less than the S-CNN, the proposed method can obtain higher fIoU than the S-CNN and other two state-of-the-art methods.

Table 6.4 Comparison of proposed method and state-of-the-art methods in terms of Bit rate saving and PSNR for surgical videos

Video	**PROI**		**MST**		**SegNet**		**S-CNN**	
	BR Saving(%)	**PSNR**	**BR Saving(%)**	**PSNR**	**BR Saving(%)**	**PSNR**	**BR Saving(%)**	**PSNR**
Z-Plasty	75	0.76	90.05	-7.89	70.18	-0.12	74.86	-0.05
Digital nerve	93.92	0.71	96.61	-15.91	77.41	-0.09	76.53	-0.06
Flexor	72.27	0.35	88.41	-10.54	65.32	-0.16	63.70	-0.01
Finger	92.87	0.18	88.15	-10.99	79.36	-0.13	82.84	-0.11
Flexor Tendom	75.92	0.70	89.78	-8.65	71.15	-0.08	75.02	-0.04
Volar wrist	98.40	2.66	95.26	-10.98	76.93	-0.14	80.02	-0.04
Arthroplasty	82.85	1.60	95.93	-14.83	74.83	-0.07	74.71	-0.05
Tendon saw injury	95.50	2.48	93.78	-7.14	82.04	-0.10	86.05	-0.02
Subcuticular	96.87	2.70	98.40	-16.56	76.31	-0.19	80.55	-0.02
Average	**87.06**	**1.34**	**92.93**	**-11.49**	**74.83**	**-0.12**	**77.14**	**-0.04**

In Table 6.4, the PROI approach is compared with the MST, SegNet and S-CNN surgical ROI segmentation techniques for surgical telementoring systems. The results indicate that the proposed method achieves 87% less bit rate with 1.34dB improvement in PSNR using PROI. This improvement is achieved for surgical video sequences with slow-moving objects.

The MST technique achieves high BR savings of 92.93% which is high compared

to our proposed technique. However, 11.49 dB of PSNR loss is observed, making it less suitable for telementoring applications. The authors in [143] use the HEVC for encoding the ROI region. The HEVC uses the RDO search process that increases the complexity. The complexity increases the coding time, which makes it unsuitable for real-time telementoring applications. The SegNet approach saved the bitrate by 75%, which is less compared to the remaining approaches. In addition, SegNet uses 26 convolutional layers in the CNN model that increases the computational complexity.

6.5 Conclusions

This paper proposed an efficient surgical telementoring system that transmits the surgical incision region at high quality with less bit rate. The surgical video consists of the surgical incision region and the background region. The background region can be removed to reduce the bit rate. The Kernelized Correlation Filter (KCF) tracker tracks the surgical incision region, crop, and writes to the video sequence. The resultant video is encoded using the SHVC video coder. SHVC uses the CNN+LSTM approach to predict the CTU structure in less time. The proposed method encodes the ROI surgical video using SHM software that saves the bit rate by 87% with a 1.34 dB improvement in video quality (PSNR).

Chapter 7

Region-Based Motion Estimation and Next-Frame Prediction using Deep Neural Network

7.1 Introduction

The rapid increase in video content generated by many multimedia devices requires a large storage space. However, the storage capacity and the bandwidth for video applications is limited, necessitating efficient video compression standard. This led to the development of the High Efficiency Video Coding (HEVC) which is more efficient than the preceding H.264 Advanced Video Coding (AVC) standard. HEVC uses advanced coding tools, such as the quad-tree structure, which divides the CTU block into CU blocks to improve efficiency. The CU has a maximum size of 64×64 pixels and a minimum size of 8×8 pixels. The Rate-Distortion (RD) cost (J), which can be estimated using equation (7.1), is used to make the split decision.

$$J = D + \lambda \times R \tag{7.1}$$

Where λ= Lagrangian multiplier, R = number of bits needed to transmit and D = Distortion.

The Distortion is estimated using the Sum of Absolute Difference (SAD) block matching technique during the Motion Estimation (ME) process. The approach for determining the motion vector is called motion estimation. The motion vector represents

the displacement between the position of the matching block in reference or the previous frame and the position of the current frame's block. Fig. 7.1 shows an example of motion vector calculation using ME.

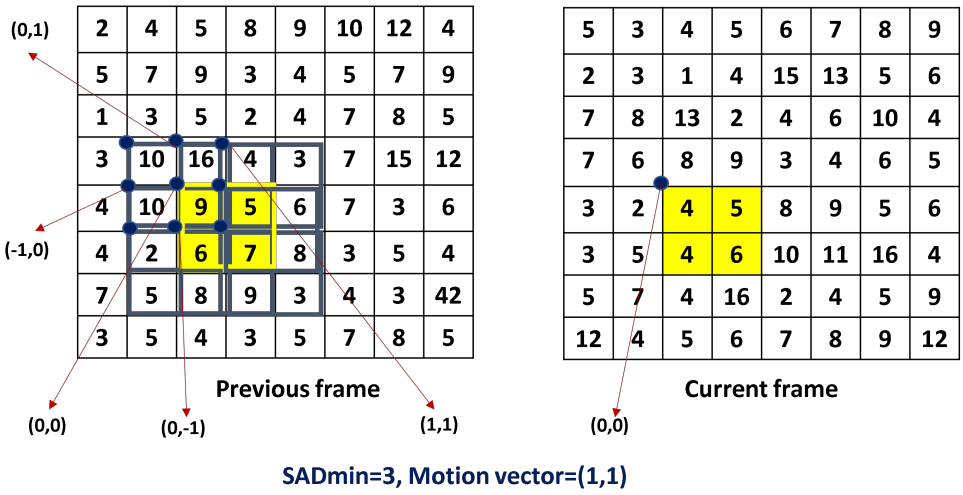


Figure 7.1 Example representation of motion estimation process

Consider the yellow color block of size 2×2. The yellow color block represents the present (or current) block of current frame and candidate block at the origin of reference frame. The SAD is performed between the current block and candidate block by using equation (7.2).

$$SAD = \sum_{q,r} |S_X(q,r) - S_Y(q,r)| \tag{7.2}$$

where $S_X(q,r)$=(q, r)th sample in X_{th} current frame block, and $S_Y(q,r)$=(q, r)th sample in Y_{th} reference frame block. Move the candidate block by one pixel towards the left, i.e., at (-1,0), and perform the SAD calculation. Similarly perform the SAD calculation between the current block and candidate block at different positions ((-1,-1), (0,-1), (1,-1), (1,0), (1,1), (0,1) and (-1,1)). The position at which the SAD_{min} is obtained is treated as a motion vector. Based on the motion vectors, the compensated frame is predicted. ME helps increase efficiency at the expense of high computational complexity, resulting in increased encoding time. The computational complexity can be decreased by directly predicting the motion compensated frame using the next-frame prediction approach.

Several authors proposed different algorithms to reduce the complexity of the ME.

Betka et al. [144] introduce a block matching approach that addresses optimization issues in a short amount of time. The developed approach contains two primary steps. Initially, the motion vectors were calculated using the Stochastic Fractal Search multi-population model. A modified fitness approximation methodology was used in the next phase to reduce computational time. Lin et al. [145] developed a Fast Predictive Search (FPS) method to accurately find the initial search point. In addition, video classification based on the motion and early termination schemes are used to reduce the search points. Wu et al. [146] improved the k-means clustering technique for estimating global motion. The speeded-up robust feature descriptor is utilized to match feature points.

The global motion vectors were derived in two dimensional feature space via homography transformation, and the next frames were stabilized using the identified global motion vectors.The researchers in [147] use the JAYA algorithm to reduce the system complexity using the search location's fitness value. Amirpour et al. [148] defined a dynamic search pattern to improve the search performance by considering the motion vector information of eight neighborhood spatial blocks. Cuevas [149] proposed a method that uses the Harmony Search optimizer fitness function to measure the matching quality of each motion vector during the ME process. The authors in [150] proposed Variable Size Block Matching with a cross-square search pattern to speed up the ME process. The above algorithms use different search patterns to speed up the motion estimation process. However, many SAD computations are required to calculate the motion vector.

In addition to the motion estimation algorithms, different next-frame prediction approaches are also proposed. Some of them are discussed below. In [151], the authors supply the input frame to the image autoencoder as a multiscale set. In addition, the convolutional kernels are extracted from the image difference using a motion autoencoder. Finally, the feature maps from the image autoencoder and the convolutional kernels are combined using cross convolution operation to predict the next frame. In [152], the state layer takes the temporal factor as an input in the autoencoder model. The input image and the time difference (Δt) of the desired prediction are provided separately to two branches of the encoder. Based on the encoder output, the decoder generates reliable frames.

The authors in [153] use GAN for the frame prediction. The generators and dis-

criminators use the multiscale structure. The loss obtained using multiple discriminators is cumulated and updated as model weights. In [154], the researchers dealt with high-dimensional sequences using a reservoir computing network. This network is used in place of the generator, which is discussed in [153]. The authors in [155] predict the human body joints in the next frame based on the structural information. The location of joints in the next frame is determined using the LSTM. The computational complexity in [156] and [157] is reduced by encoding the entire input video sequence and unwinding using the decoder for the multiple predictions. In [158], gated autoencoders predict the next frame and use recurrent connections to forecast any length sequence. In [159], the frames and the human actions are encoded separately. The output features of these encoders are combined and given as input to the LSTM for the next-frame prediction. These techniques generates the next-frame with low quality.

In this work, we have proposed region-based motion estimation using the YOLOv4 algorithm. The motion vector is calculated using the centroid point obtained from the bounding box coordinates. The bounding box regions are detected using the YOLOv4 algorithm. Moreover, the next-frame prediction model using ConvLSTM layers is proposed that predicts the next frame based on the input five frames.

7.2 Proposed Work

In this section two different methods are proposed, which can be used in HEVC to reduce the computation complexity of the encoder. The two approaches are region-based motion estimation using YOLOv4 and the next-frame prediction using Recurrent Neural Network.

7.2.1 Region-based motion estimation using YOLOv4

This method determines the motion vector for each region in the frame using the YOLOv4 algorithm. The proposed framework in Fig. 7.2 involves splitting the video sequence into frames, detecting the bounding box for each region in frame using YOLOv4, finding the centroid point of each region using the bounding box coordinates, and deter-

mining the motion vector based on the centroid point.

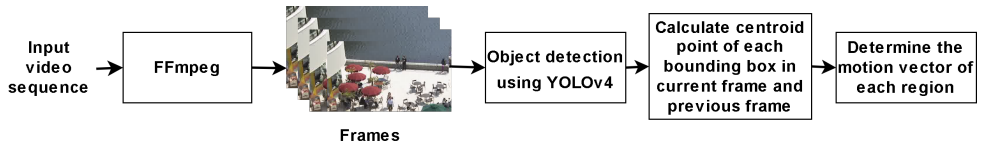


Figure 7.2 Framework to determine motion vectors using YOLOv4

Initially, the video sequence is split into frames using FFmpeg. FFmpeg is a set of tools and libraries for working with multimedia files like video, audio, subtitles, and metadata. The command to convert the video sequence into frames is given below.

ffmpeg -i input.mp4 -vf fps=25 image.jpg

The above command converts the mp4 video sequence into JPEG frames. The output frames are given as input to the object detector. The YOLOv4 object detector shown in Fig. 7.3 is used to detect bounding boxes of each region in the current and previous frames. YOLOv4 consists of CSPDarknet-53 to extract features of the input frame through five Residual blocks (Res1-Res5). This network contains 53 convolutional layers with 1×1 and 3×3 sizes, and each layer is connected to batch normalization and the Mish activation layers. The Spatial Pyramid Pooling networks (SPPnet) significantly increases the model's receptive field through distinct max-pooling layers with sizes of 5, 9, and 13. The Path Aggregation Network (PANet) repeatedly extracts the features using the top-down and the bottom-up techniques. Three YOLO heads Y1 with 19×19, Y2 with 38×38 and Y3 with 76×76 are employed to fuse and engage with feature maps of various scales to detect the objects of various sizes.

The loss in YOLOv4 is given as

$$loss = -\lambda_1 \sum (O_i ln(p_i) + (1 - O_i) ln(1 - p_i)) \\ - \lambda_2 \sum_{i \epsilon Box} \sum_{k \epsilon class} (Q_{ik} ln(p_{ik}) + (1 - Q_{ik}) ln(1 - p_{ik})) + \lambda_3 (1 - IOU + \frac{d^2(A_c, B_c)}{l^2} + \alpha v) \tag{7.3}$$

Where O_i indicates whether the predicted bounding box 'i' contains the object, P_i gives the likelihood that the prediction box contains an actual object. Q_{ik} and p_{ik} gives whether the k-class object and the probability present in the bounding box 'i'. YOLOv4

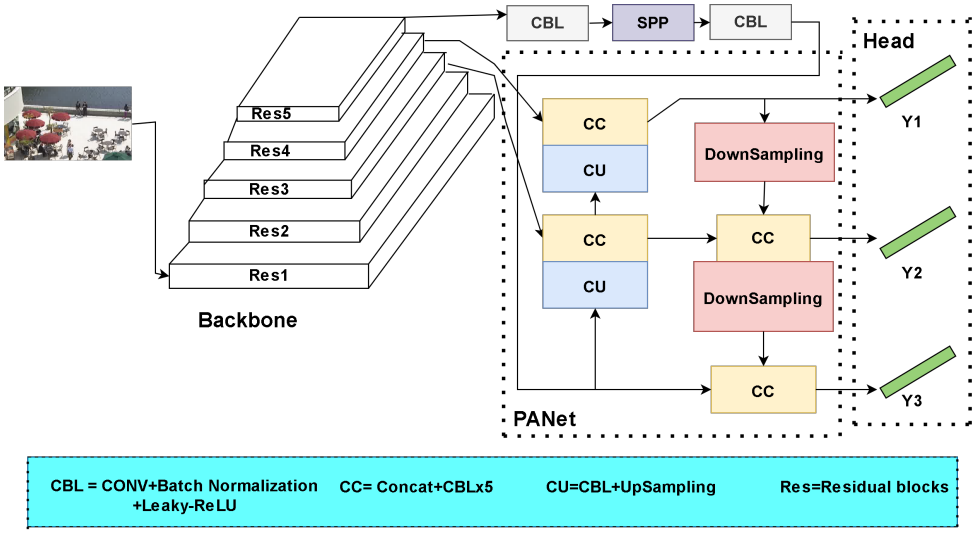


Figure 7.3 Architecture of YOLOv4

uses the Complete Intersection Over Union (CIOU) technique [160] for calculating the object localization offset loss, where αv is the aspect ratio and (A_c,B_c) is the center point.

The centroid is calculated from the bounding box coordinates after detecting the objects in the current frame and the previous frames. Finally, for each region detected in the current frame, the motion vector is calculated based on the centroid location of the corresponding region bounding box in the previous frame.

7.2.2 Next-frame prediction using Recurrent Neural Network

This section proposes the next-frame prediction method to predict the sixth frame based on the first five frames. The proposed technique is shown in Fig. 7.4, consisting of layers like Reshape, Permute, Gaussian Noise, Lambda, ConvLSTM2D, and Conv2D layers. Initially, the five frames of the sequence are given as input. Each frame is reshaped to 96×96 using the Reshape layer. The output is passed to the permute layer to interchange the dimensions according to the given pattern. The permuted output is given to the gaussian noise layer to mitigate the overfitting problem. After the gaussian noise layer, the lambda layer is employed for simple stateless computations. Moreover, the permute layer takes the output of the gaussian noise layer and changes the dimensions to (96,96,5,3). The permuted output is passed through two sets of ConvLSTM2D, Dropout, and Upsampling2D layers.

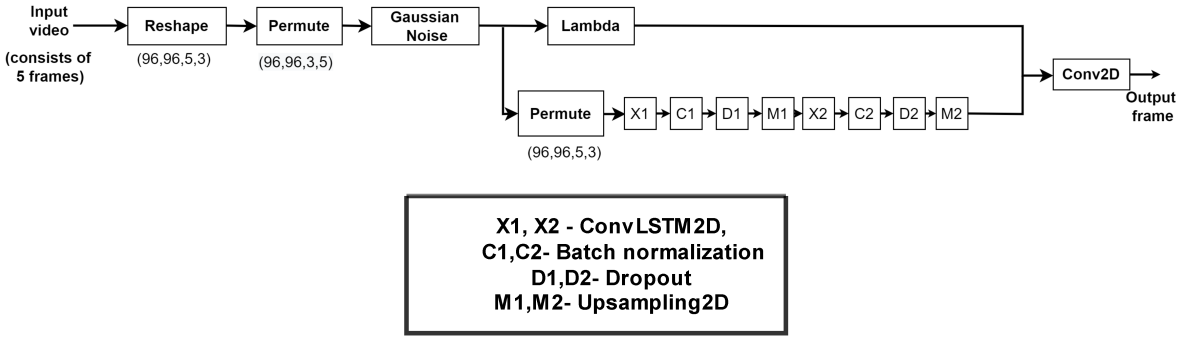


Figure 7.4 Next-frame prediction model

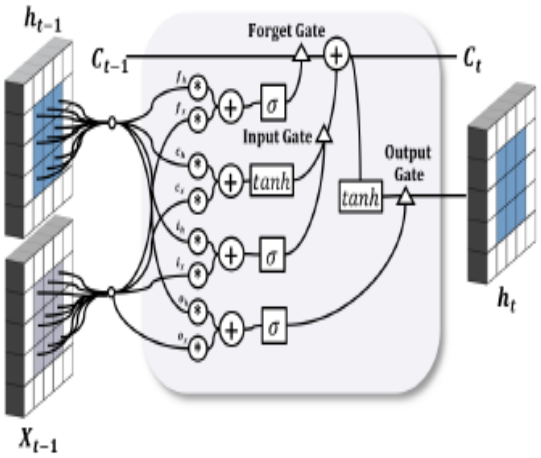


Figure 7.5 ConvLSTM cell internal structure

ConvLSTM is a kind of LSTM (Long Short Term Memory) where the convolution is performed inside the LSTM cell. It helps to learn the long-term dependencies between the images. The ConvLSTM2D structure is shown in Fig. 7.5. LSTM consists of forget gate, output gate, and the input gate. ConvLSTM uses convolution operation in place of matrix multiplication operation at each gate of the LSTM. LSTM takes a one-dimensional vector as input, whereas ConvLSTM takes the 3D data as input. The ConvLSTM model is formulated using the following equations.

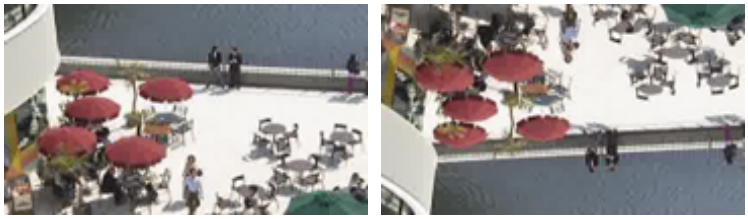
$$i_t = \sigma(b_i + W_{hi} * h_{t-1} + W_{xi} * X_t) \tag{7.4}$$

$$f_t = \sigma(b_f + W_{hf} * h_{t-1} + W_{xf} * X_t) \tag{7.5}$$

$$o_t = \sigma(b_o + W_{ho} * h_{t-1} + W_{xo} * X_t) \tag{7.6}$$

$$C_t = i_t \circ tanh(b_c + W_{xc} * X_t + h_{t-1} * W_{hc}) + f_t \circ C_{t-1} \tag{7.7}$$

$$h_t = o_t \circ tanh(C_t) \tag{7.8}$$



(a) Original **(b)** Augmented

Figure 7.6 Original and Augmented frames of BQSquare video sequence

Where f_t, i_t and o_t are the outputs at time 't' of forget gate, input gate and output gate, respectively. C_t represents output of cell and h_t is the cell hidden state. '*' and '$\circ$' represents convolution and Hadamard product operation.

Batch Normalisation uses a transformation to keep the output standard deviation and mean close to 1 and 0, respectively. Upsampling2D layer helps to increase the dimension of the array. It is the opposite of pooling, where it repeats rows and columns of the input.

Finally, the Upsampling2D and Lambda layer output is passed through the Conv2D to get the output frame.

7.3 Experimental Results

In this section, the experimental analysis is done seperately for two approaches. In section 7.3.1, experimental results for region-based motion estimation is analyzed and section 7.3.2 discusses the experimental results of the next-frame prediction approach.

7.3.1 Experimental analysis of region-based motion estimation

The proposed method is simulated on windows machine using Intel Core i7-7500U processor. Initially, the BQSquare video sequence is split into frames using the FFmpeg software at the rate of 60fps. Total 600 frames are generated, out of which the regions in 400 frames are labelled and each labelled frame undergoes augmentation to increase the number of frames.

The labeled frames are used for training the YOLOv4. The original and augmented

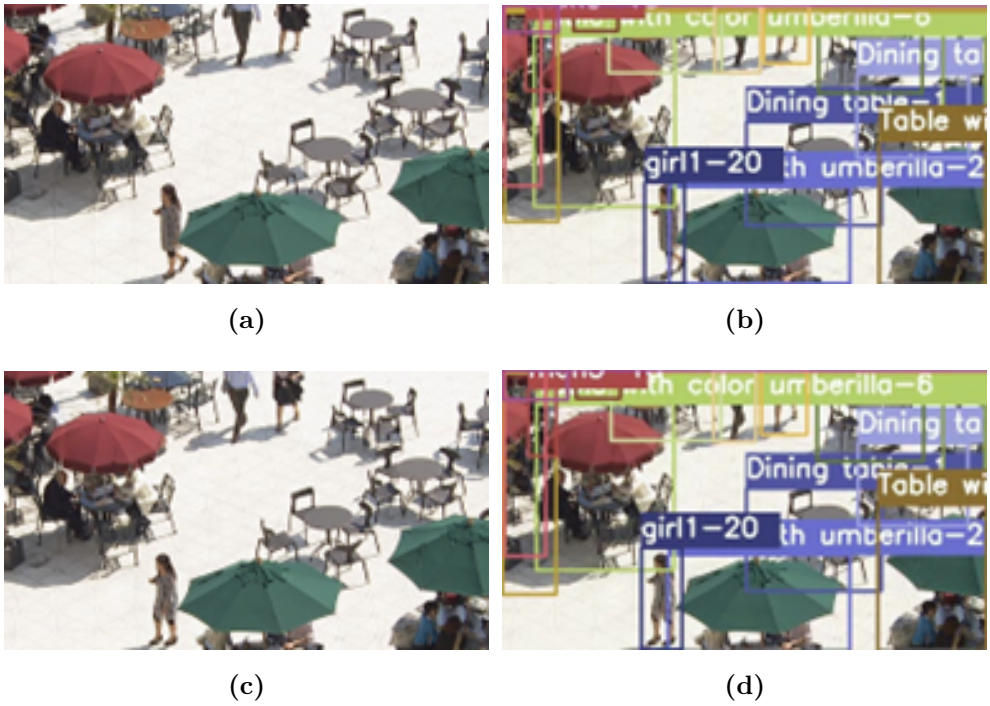


Figure 7.7 (a) and (c) represents the original frames 3 and 4 of BQSquare, (b) and (d) represents the output frames 3 and 4 of BQSquare with bounding boxes detected using YOLOv4

frames are shown in Fig. 7.6. The YOLOv4 is trained for 44 different classes. During training, YOLOv4 is configured with the maximum batches of 88000; batch size and subdivisions are 64 and 32, respectively. After training, the inference is performed on the BQSquare sequence with untrained frames. The original and the object detected frames are shown in Fig. 7.7.

Consider frame 3 as a previous frame and frame 4 as a current frame. The bounding box coordinates along with object ids for different regions in frame 3 and frame 4 are given in Table 7.1. Table 7.1 shows that the BBox coordinates are the same in two frames for some of the classes like Dining table1, Dining table3, Table with umbrella1, Table with color umbrella1, Lady2, Man2, base, man3 and Table with color umbrella classes. The same coordinates in two frames represent that the class remains static and observed no movement between the frames. The center of the bounding box (X_{cen},Y_{cen})can be calculated by using equation (7.9).

$$(X_{cen}, Y_{cen}) = (\frac{(x_1 + x_2)}{2}, \frac{(y_1 + y_2)}{2}) \tag{7.9}$$

Where (x_1,y_1) represents the top-left coordinate of bounding box and (x_2,y_2) represents the bottom-right coordinate of bounding box. The centroid for different classes in frame three and frame four is given in Table 7.2. Finally, the motion vector is determined which is the displacement between the centroid point of each class in frame 4 and the centroid location of the corresponding class in frame 3.

Table 7.1 Bounding box coordinates of different classes in frame 3 and frame 4 of BQSquare sequence

Tracker ID	Class	Frame 3 BBox Coordinates	Frame 4 BBox Coordinates
1	Dining table	208, 95, 326, 187	210, 96, 324, 186
2	Table with umbrella	137, 151, 300, 239	138, 151, 300, 239
3	Dining table2	302, 54, 404, 128	302, 54, 403, 128
4	Dining table1	272, 0, 363, 72	272, 0, 363, 72
5	Dining table3	380, 13, 414, 83	380, 13, 414, 83
6	Table with color umbrella	21, 24, 145, 175	23, 24, 146, 174
7	table with color	88, 0, 184, 50	88, 0, 184, 51
8	Table with umbrella1	325, 113, 415, 240	325, 113, 415, 240
9	Table with color umbrella1	0, 6, 46, 183	0, 6, 46, 183
10	Lady2	236, 0, 264, 40	236, 0, 264, 40
11	Man2	188, 0, 233, 39	188, 0, 233, 39
12	base	58, 0, 94, 16	58, 0, 94, 16
13	man3	17, 4, 38, 72	17, 4, 38, 72
14	Table with color umbrella	0, 0, 24, 143	0, 0, 24, 143

Complexity analysis of proposed and State-of-the-art methods

The authors in [47, 49, 50] uses block-based ME during encoding process. Consider frame 3 of 416×240 pixels in size, as shown in Fig. 7.7 (a). The frame is divided into 16×16 equal chunks and finds the matching block in the reference frame using a 4-point diamond search pattern for each block, as shown in Fig. 7.8. The 4-point diamond search pattern is chosen as an example to evaluate the complexity of the motion estimation process. For each 16×16 block SAD computation, 256 subtractions and 255 additions are required. Hence for finding the motion vector for each block, five times SAD must be performed, which increases the complexity. The complexity is further increased when the motion vector is calculated for all the blocks in the frame.

Table 7.2 Centroid of bounding boxes of different classes in Frame 3 and frame 4 of BQSquare sequence

Tracker ID	Class	Centroid in frame 3	Centroid in frame 4	Motion Vector
1	Dining table	(267,141)	(267,141)	(0,0)
2	Table with umbrella	(218.5,195)	(219,195)	(-0.5,0)
3	Dining table2	(353,91)	(352.5,91)	(0.5,0)
4	Dining table1	(317.5,36)	(317.5,36)	(0,0)
5	Dining table3	(397,48)	(397,48)	(0,0)
6	Table with color umbrella	(83,99.5)	(84.5,99)	(-1.5,0.5)
7	Table with color	(136,25)	(136,25.5)	(0,-0.5)
8	Table with umbrella1	(370,176.5)	(370,176.5)	(0,0)
9	Table with color umbrella1	(23,94.5)	(23,94.5)	(0,0)
10	Lady2	(250,20)	(250,20)	(0,0)
11	Man2	(210.5,19.5)	(210.5,19.5)	(0,0)
12	base	(76,8)	(76,8)	(0,0)
13	man3	(27.5,38)	(27.5,38)	(0,0)
14	Table with color umbrella	(12,71.5)	(12,71.5)	(0,0)

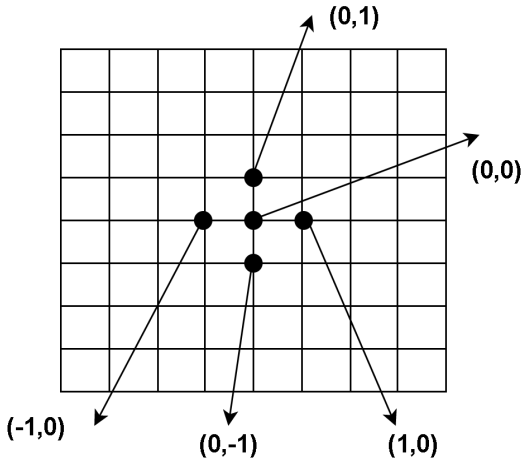


Figure 7.8 Example frame with 4-point diamond search pattern

However, using the proposed method, the regions or objects are detected directly by using the YOLOv4 algorithm. In frame 3, only 14 regions need to be detected to find the motion vectors in the frame. The proposed method is compared to state-of-the-art algorithms: Modified Diamond Search Algorithm (MDSA), Cross Diamond Search (CDS)

Table 7.3 Comparison results of the proposed technique and the state-of-the-art methods in terms of computation time

Frame No.	MDSA	CDS	DS	Prop
2	36.32	31.19	36.53	1.02
3	36.19	31.26	36.32	1.31
4	35.43	31.70	35.24	1.09
5	36.26	30.19	36.43	1.44
6	35.31	30.72	36.25	1.32
7	35.92	31.91	36.71	1.33
8	36.12	31.10	36.34	1.49
9	36.31	31.34	35.66	1.53
10	36.03	30.91	34.21	1.09

and Diamond Search (DS) algorithms in Table 7.3. The results show that the proposed method decreases the computation time significantly compared to the state-of-the-art approaches.

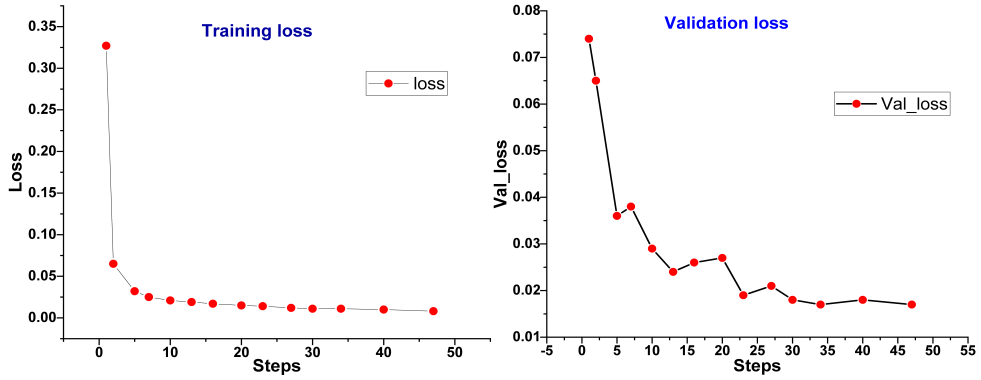
7.3.2 Experimental analysis of next-frame prediction model

In this approach, UCF 101 dataset is used, which contains sports as well as daily life scenes. The proposed model is trained using the images of the ApplyEyeMake video sequence present in the UCF101 dataset. The sequence has a resolution of 320×240 with 25 fps. A total of 2000 frames are considered, and each frame is reshaped to 96×96. Out of 2000 frames, 1604 frames are used for training, and the rest are used for testing. Similarly, the BQSquare HEVC video sequence frames are reshaped to 96×96 and are used for training the next frame prediction model. The hyperparameters for the simulation of the proposed model are given in Table 7.4.

The performance of the proposed method is evaluated using Structural Similarity Index (SSIM), PSNR and Mean Square Error (MSE), which are given in equations (7.10),

Table 7.4 Experimental parameters

Hyperparameters	
Optimizer	Adam
Loss	Mean Square Error
Metric	Perceptual distance
Batch size	32
Height	96
Width	96



(a) **(b)**

Figure 7.9 Training and validation loss curves of BQSquare HEVC video sequence

(7.11), and (7.12).

$$SSIM = \frac{(2\mu_Y\mu_{\hat{Y}} + P_1) + (2\sigma_{Y\hat{Y}} + P_2)}{(\mu_Y^2 + \mu_{\hat{Y}}^2 + P_1)(\sigma_Y^2 + \sigma_{\hat{Y}}^2 + P_2)} \tag{7.10}$$

$$PSNR = 10log_{10}\frac{\hat{Y}_{max}^2}{\sum_{i=1}^{T}(Y_i - \hat{Y}_i)^2} \tag{7.11}$$

$$MSE = \frac{1}{T}\sum_{i=1}^{T}(Y_i - \hat{Y}_i)^2 \tag{7.12}$$

Where,

Y and $\hat{Y}\rightarrow$ ground truth and predicted values

T$\rightarrow$ Total number of pixels,

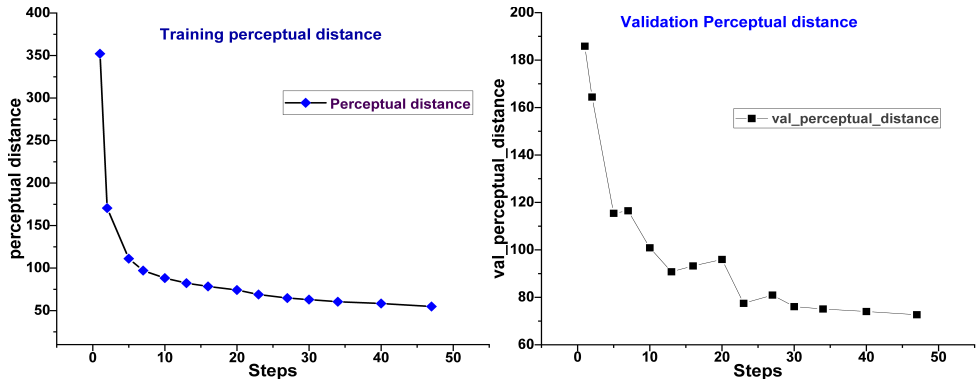
$\hat{Y}_{max} \rightarrow$ maximum possible image intensity value,

σ_Y and $\mu_Y \rightarrow$ variance and average of Y,

$\sigma_{\hat{Y}}$ and $\mu_{\hat{Y}} \rightarrow$ variance and average of $\hat{Y}$.

Table 7.5 Comparison results of the proposed technique and the state-of-the-art methods

UCF101			
Approach	**MSE**	**PSNR**	**SSIM**
[**159**]	0.008	20.44	0.613
[**161**]	0.009	23.36	0.674
[**158**]	0.054	21.03	0.632
[**155**]	0.011	22.81	0.641
Prop	**0.003**	**29.35**	**0.880**



(a) (b)

Figure 7.10 Training and validation perceptual distance curves of BQSquare HEVC video sequence

The proposed model is compared to state-of-the-art approaches in [159], [161], [158] and [155]. The results in Table 7.5 show that the proposed approach achieves SSIM of 0.880 and PSNR of 29.35dB at the cost of MSE loss of 0.003. Our method outperforms the remaining approaches in terms of PSNR and SSIM values.

Another metric, such as the perceptual distance, is used to analyze the performance. Fig. 7.9 (a) and (b) shows the training and validation loss of the proposed method for the BQSquare video sequence. The loss curves depict that the loss is as low as 0.02, representing that the proposed method is trained efficiently. Fig. 7.10(a) and (b) show that the perceptual distance is considerably decreased. This showcases that the predicted

Figure 7.11 Input frames of the ApplyEyeMake video sequence

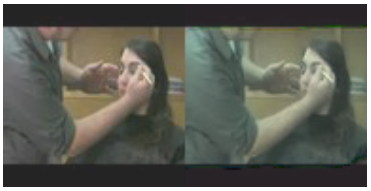


Figure 7.12 Output sixth frame of the ApplyEyeMake video sequence

frame almost resembles the original frame.

Figure 7.13 Input frames of the BQSquare HEVC video sequence

Figure 7.14 Output sixth frame of the BQSquare HEVC video sequence

The five input frames of the ApplyEyeMake and BQSquare sequences are shown in Fig. 7.11 and Fig. 7.13. The ground truth and the generated output of the ApplyEyeMake and BQSquare sequences are shown in Fig. 7.12 and Fig. 7.14. The output images show that the predicted frame looks like the original frame.

7.4 Conclusions

In this work, the region-based motion estimation is performed using the YOLOv4 algorithm. This algorithm detects the bounding box of the regions present in the frame. Then the centroid point is calculated for each bounding box in the current and reference frames. Finally, the motion vector is calculated based on the object class and centroid point. The proposed method significantly decreases the complexity of the motion estimation process compared to the state-of-the-art techniques.

In addition, the next-frame prediction model is proposed that helps to predict the next frame by taking the first five frames as input. This model can directly predict the motion compensated frame by skipping the computational complex ME process in HEVC. This model is tested on the UCF101 dataset video sequence and the HEVC video sequence. The results show that the model predicts the next frame efficiently with high PSNR and low MSE values compared to the state-of-the-art techniques.

Chapter 8

Conclusions and Future Scope

This chapter concludes the thesis by underlining the main contributions. It also presents the possible directions of future work.

8.1 Conclusions

In HEVC, an advanced coding tool such as a quad-tree CTU structure is used to improve efficiency at the cost of computational complexity. The complexity increases due to the RDO search process determining the optimal CTU partitions. Moreover, the motion estimation process is used to determine the motion vectors. The computational complexity of the recursive RDO search process and motion estimation increases the encoding time. In this thesis, different approaches are proposed to decrease the encoding time. The contributions of the research work are concluded as follows.

Chapter 3 presents the MLRVS algorithm and the complexity reduction method to reduce the encoder's encoding time. The complexity reduction algorithm is used to detect the skip mode early. The multi-resolution frame structure is created using Vertical Subsampling to reduce the SAD computations and to obtain the motion vector with global minima. It also helps to overcome the local minima problem. Moreover, different search patterns like NCDD and NCDH are used to accelerate the search process to find the motion vector. The results show that the encoding time of the encoder is decreased by 55% with the MLRVS algorithm using the NCDD search pattern and 56% with MLRVS using the NCDH search pattern compared to the HM16.5 standard.

Chapter 4 proposes the CU size prediction method using a CNN deep learning methodology to reduce the encoding time. The CNN approach is used in place of the RDO search process, which uses spatial features to predict the CU size. In addition, the MRCDO method enhances the speed of the motion estimation process. The MRCDO motion estimation technique uses the CDO search pattern to speed up the motion vector calculation. Compared to the HM-16.5 standard, the suggested solution reduces encoding time by 66.91 percent on average, with negligible degradation in quality.

In chapter 5, SHVC is used to encode the video sequences. SHVC uses more than one layer to provide spatial scalability. However, the complexity increases significantly compared to HEVC due to the many layers. Hence, SHVC using an Early Terminated CNN+LSTM structure is developed to predict the partitions of CTU. The CNN+LSTM approach considers spatial and temporal features to predict the CTU partitions. The proposed method achieves 53% savings in encoding time for LDP configuration and 59% for RA configuration, which is significantly higher than state-of-the-art methods.

Chapter 6 develops an efficient surgical telementoring system that transmits the surgical incision region at high quality with less bit rate. The surgical incision region in the video sequence is identified by the Kernelized Correlation Filter (KCF) tracker. The tracker tracks the surgical incision region, then crops and writes to the video sequence. The resultant video is encoded using the SHVC video coder. SHVC uses the CNN+LSTM model to encode the video with less complexity. The proposed method saves the bit rate by 87% with good video quality (PSNR).

Finally, in chapter 7, we proposed the region-based motion estimation using YOLOv4. This model helps to detect the regions present in the frame. Based on the bounding box coordinates of the region, the centroid point is calculated, which helps to find the motion vectors fastly. The results prove that the proposed method takes less time than the state-of-the-art approaches to determine the motion vectors of the frame.

We also proposed the next frame prediction using a deep neural network to reduce the encoding time in HEVC. This technique uses the ConvLSTM layer to predict the next frame based on the previous five frames. The proposed approach can be used in HEVC to predict the motion-compensated frame by skipping the motion estimation and compensation process. The output figures represent that the next frame can be predicted

accurately based on the previous frames.

8.2 Future Scope

The work proposed in this thesis can be extended for future research. Some of the possible directions in which the problems can be further pursued are:

- This thesis uses machine learning approaches to predict the CTU partitions in HEVC. However, this work can be extended by using the same approaches to predict the asymmetric partition modes present in Inter prediction.
- In HEVC, the motion estimation process consumes more time. Hence, developing Hardware accelerators for motion estimation algorithms can decrease the encoding time.
- The computational complexity of HEVC is increased due to the quad-tree TU partitions. Therefore, developing the machine learning model to predict the TU partitions decreases the complexity.
- Further, the current work can be extended by developing the hardware architecture for the Motion Compensation block that helps to generate the motion compensated frames.

Publications

List of International Journals:

1. S. Karthik Sairam, P. Muralidhar "A motion estimation based algorithm for encoding time reduction in HEVC" Defence Science Journal, Vol. 72, No. 1, January 2022, pp. 56-66, DOI : 10.14429/dsj.72.16733 **(SCI)**

2. S. Karthik Sairam, P. Muralidhar " Object Tracking Based Surgical Incision Region Encoding using Scalable High Efficiency Video Coding for Surgical Telementoring Applications " Radioengineering Journal, Vol. 31, No. 2, June 2022, pp. 231-242, DOI : 10.13164/re.2022.0231**(SCI)**

3. S. Karthik Sairam, P. Muralidhar "A Deep Learning Approach in Scalable High Efficiency Video Coding for fast Coding Unit size decision", IETE Technical Review journal, 2022, pp. 1-16, DOI: 10.1080/02564602.2022.2100492 **(SCI)**

4. S. Karthik Sairam, P. Muralidhar "Fast Convolutional Neural Network based Coding Unit Size Prediction in HEVC", *Multidimensional Systems and Signal Processing Journal*. **(Under Review) (SCI)**

List of International Conferences:

1. S. Karthik Sairam, P. Muralidhar," Hybrid Fast Motion Estimation for HEVC",6th International Conference on Signal Processing and Integrated Networks (SPIN),2019.

2. S. Karthik Sairam, P. Muralidhar,"Fast encoding in HEVC using subsampling with unsymmetrical octagonal search pattern", IEEE 16th India Council International Conference (INDICON),2019.

3. S. Karthik Sairam, P. Muralidhar,"Fast Encoding using X-Search Pattern and Coded Block Flag Fast Method" First International Conference Communications, Signal Processing and VLSI (IC2SV 2019) held at the National Institute of Technology Warangal,2019.

Bibliography

[1] T. TK, M. Marta, B. Vittorio, and R. Naeem, "Report on HEVC compression performance verification testing," *JCT-VC*, 2014.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[4] T. TK, M. Marta, B. Vittorio, and R. Naeem, "Report on HEVC compression performance verification testing," *JCT-VC*, 2014.

[5] J. Hsieh, J. Cai, Y. Wang, and Z. Guo, "ML-Assisted DVFS Aware HEVC Motion Estimation Design Scheme for Mobile APSoC," *IEEE Syst. J.*, vol. 13, no. 4, pp. 4464–4473, 2019.

[6] N. C. Vayalil, M. Paul, and Y. Kong, "A Residue Number System Hardware Design of Fast-Search Variable-Motion-Estimation Accelerator for HEVC/H.265," *IEEE Trans.Circuits Syst. Video Technol.*, vol. 29, no. 2, pp. 572–581, 2019.

[7] G. Cebrián-Márquez, J. L. Martínez, and P. Cuenca, "A Motion-Based Partitioning Algorithm for HEVC Usinga Pre-Analysis Stage," *IEEE Trans.Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1448–1461, 2019.

[8] K. Fan, R. Wang, G. Li, and W. Gao, "Efficient Prediction Methods With Enhanced Spatial-Temporal Correlation for HEVC," *IEEE Trans.Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3716–3728, 2019.

[9] F. Pakdaman, M. Hashemi, and M. A. Ghanbari, "low complexity and computationally scalable fast motion estimation algorithm for HEVC," *Multimed. ToolsAppl.*, vol. 79, p. 11639–11666, 2020.

[10] T. Jiang, X.and Song, T. Katayama, and J.-S. Leu, "Spatial Correlation- Based Motion-Vector Prediction for Video-Coding Efficiency Improvement," *Symmetry*, vol. 11, no. 2, p. 129, 2019.

[11] S. Gogoi and R. Peesapati, " A hybrid hardware orientedmotion estimation algorithm for HEVC/H.265," *J. Real-Time Image Proc.*, vol. 18, p. 953–966, 2021.

[12] F. Luo, S. Wang, S. Wang, X. Zhang, S. Ma, and W. Gao, " GPU-Based Hierarchical Motion Estimation for High Efficiency Video Coding," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 851–862, 2019.

[13] S. Y. Jou, S. J. Chang, and T. S. Chang, "Fast Motion Estimation Algorithm and Design for Real Time QFHD High Efficiency Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 9, pp. 1533–1544, 2015.

[14] K. Singh and S. Rafi Ahamed, "Low Power Motion Estimation Algorithm and Architecture of HEVC/H.265 for Consumer Applications," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 3, pp. 267–275, 2018.

[15] F. Cheng, T. Tillo, J. Xiao, and B. Jeon, "Texture Plus Depth Video Coding Using Camera Global Motion Information," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2361–2374, 2017.

[16] S.-H. Park and J.-W. Kang, "Fast Affine Motion Estimation for Versatile Video Coding (VVC) Encoding," *IEEE Access*, vol. 7, pp. 158 075–158 084, 2019.

[17] L. Trudeau, S. Coulombe, and C. Desrosiers, "Cost-Based Search Ordering for Rate-Constrained Motion Estimation Applied to HEVC," *IEEE Transactions on Broadcasting*, vol. 64, no. 4, pp. 922–932, 2018.

[18] Z. Pan, J. Lei, Y. Zhang, X. Sun, and S. Kwong, "Fast Motion Estimation Based on Content Property for Low-Complexity H.265/HEVC Encoder," *IEEE Transactions on Broadcasting*, vol. 62, no. 3, pp. 675–684, 2016.

[19] H.-S. Kim, J.-H. Lee, C.-K. Kim, and B.-G. Kim, "Zoom Motion Estimation Using Block-Based Fast Local Area Scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 9, pp. 1280–1291, 2012.

[20] G. He, D. Zhou, Y. Li, Z. Chen, T. Zhang, and S. Goto, "High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 12, pp. 3138–3142, 2015.

[21] S. Gogoi and R. Peesapati, "Design and Implementation of an Efficient Multi-Pattern Motion Estimation Search Algorithm for HEVC/H.265," *IEEE Transactions on Consumer Electronics*, vol. 67, no. 4, pp. 319–328, 2021.

[22] L. Jia, C.-Y. Tsui, O. C. Au, and K. Jia, "A New Rate-Complexity-Distortion Model for Fast Motion Estimation Algorithm in HEVC," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 835–850, 2019.

[23] N. Hu and E.-H. Yang, "Fast Motion Estimation Based on Confidence Interval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 8, pp. 1310–1322, 2014.

[24] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An Effective CU Size Decision Method for HEVC Encoders," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 465–470, 2013.

[25] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, 2011.

[26] G. Correa, P. Assuncao, L. Agostini, and L. A. D. S. Cruz, "Coding Tree Depth Estimation for Complexity Reduction of HEVC," in *2013 Data Compression Conference*, 2013, pp. 43–52.

[27] J. H. Bae and M. H. Sunwoo, "Adaptive Early Termination Algorithm Using Coding Unit Depth History in HEVC," *Journal of Signal Processing Systems*, vol. 91, p. 863–873, 2019.

[28] L. Shen, Z. Zhang, and P. An, "Fast CU size decision and mode decision algorithm for HEVC intra coding," *IEEE Transactions on Consumer Electronics*, vol. 59, no. 1, pp. 207–213, 2013.

[29] C. Yue-Feng, W. Wan-Liang, and Y. Xin-Wei, "A fast CU depth decision mechanism for HEVC," *Information Processing Letters*, vol. 115, no. 9, pp. 719–724, 2015.

[30] L. Zhuoming, Z. Yu, D. Zheng, R. Kanza, C. Yaohui, X. Zhenjian, and Y. Wenchao, "A fast CU partition method based on CU depth spatial correlation and RD cost characteristics for HEVC intra coding," *Signal Processing: Image Communication*, vol. 75, pp. 141–146, 2019.

[31] S. Huade, L. Fan, and C. Huanbang, "A fast CU size decision algorithm based on adaptive depth selection for HEVC encoder," in *2014 International Conference on Audio, Language and Image Processing*, 2014, pp. 143–146.

[32] Z. Pan, S. Kwong, Y. Zhang, J. Lei, and H. Yuan, "Fast Coding Tree Unit depth decision for high efficiency video coding," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 3214–3218.

[33] X. Liu, Y. Li, D. Liu, P. Wang, and L. T. Yang, "An Adaptive CU Size Decision Algorithm for HEVC Intra Prediction Based on Complexity Classification Using Machine Learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 1, pp. 144–155, 2019.

[34] B. Erabadda, T. Mallikarachchi, G. Kulupana, and A. Fernando, "iCUS: Intelligent CU Size Selection for HEVC Inter Prediction," *IEEE Access*, vol. 8, pp. 141 143–141 158, 2020.

[35] T. Mallikarachchi, D. S. Talagala, H. K. Arachchi, and A. Fernando, "Content-Adaptive Feature-Based CU Size Prediction for Fast Low-Delay Video Encoding in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 3, pp. 693–705, 2018.

[36] Y.-T. Kuo, P.-Y. Chen, and H.-C. Lin, "A Spatiotemporal Content-Based CU Size Decision Algorithm for HEVC," *IEEE Transactions on Broadcasting*, vol. 66, no. 1, pp. 100–112, 2020.

[37] B. Erabadda, T. Mallikarachchi, C. Hewage, and A. Fernando, "Quality of Experience (QoE)-Aware Fast Coding Unit Size Selection for HEVC Intra-Prediction," *Future Internet*, 2019.

[38] L. Shen, Z. Zhang, and Z. Liu, "Adaptive Inter-Mode Decision for HEVC Jointly Utilizing Inter-Level and Spatiotemporal Correlations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 10, pp. 1709–1722, 2014.

[39] S. Liquan, Z. Zhang, and Z. Liu, "Effective CU Size Decision for HEVC Intracoding," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4232–4241, 2014.

[40] B. Min and R. C. C. Cheung, "A Fast CU Size Decision Algorithm for the HEVC Intra Encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 892–896, 2015.

[41] J. Xiong, H. Li, Q. Wu, and F. Meng, "A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 559–564, 2014.

[42] S. Bouaafia, R. Khemiri, F. Sayadi, and M. Atri, "Fast CU Partition-Based Machine Learning Approach for Reducing HEVC Complexity," *Journal of Real-Time Image Processing*, 02 2020.

[43] B. Huang, Z. Chen, Q. Cai, M. Zheng, and D. O. Wu, "Rate-Distortion-Complexity Optimized Coding Mode Decision for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 3, pp. 795–809, 2020.

[44] Y. Lu, X. Huang, H. Liu, Y. Zhou, H. Yin, and L. Shen, "Hierarchical Classification for Complexity Reduction in HEVC Inter Coding," *IEEE Access*, vol. 8, pp. 41 690–41 704, 2020.

[45] R. R. Sharma and K. V. Arya, "Parameter optimization for HEVC/H.265 encoder using multi-objective optimization technique," in *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, 2016, pp. 592–597.

[46] S. Yan, L. Hong, W. He, and Q. Wang, "Group-Based Fast Mode Decision Algorithm for Intra Prediction in HEVC," in *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*, 2012, pp. 225–229.

[47] A. Jiménez-Moreno, E. Martínez-Enríquez, and F. Díaz-de María, "Complexity Control Based on a Fast Coding Unit Decision Method in the HEVC Video Coding Standard," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 563–575, 2016.

[48] K.-Y. Kim, H.-Y. Kim, J.-S. Choi, and G.-H. Park, "MC Complexity Reduction for Generalized P and B Pictures in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 10, pp. 1723–1728, 2014.

[49] H. Lee, H. J. Shim, Y. Park, and B. Jeon, "Early Skip Mode Decision for HEVC Encoder With Emphasis on Coding Quality," *IEEE Transactions on Broadcasting*, vol. 61, no. 3, pp. 388–397, 2015.

[50] S. Ahn, B. Lee, and M. Kim, "A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 422–435, 2015.

[51] P. Nalluri, L. N. Alves, and A. Navarro, "Complexity reduction methods for fast motion estimation in HEVC," *Signal Processing: Image Communication*, vol. 39, pp. 280–292, 2015.

[52] D. Wang, Y. Sun, J. Liu, F. Dufaux, X. Lu, and B. Hang, "Probability-Based Fast Intra Prediction Algorithm for Spatial SHVC," *IEEE Transactions on Broadcasting*, vol. 68, no. 1, pp. 83–96, 2022.

[53] H. Yang, L. Shen, X. Dong, Q. Ding, P. An, and G. Jiang, "Low-Complexity CTU Partition Structure Decision and Fast Intra Mode Decision for Versatile Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1668–1682, 2020.

[54] X. Dong, L. Shen, M. Yu, and H. Yang, "Fast Intra Mode Decision Algorithm for Versatile Video Coding," *IEEE Transactions on Multimedia*, vol. 24, pp. 400–414, 2022.

[55] Z. Liu, T.-L. Lin, and C.-C. Chou, "Efficient prediction of CU depth and PU mode for fast HEVC encoding using statistical analysis," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 474–486, 2016.

[56] H. R. Tohidypour, M. T. Pourazad, and P. Nasiopoulos, "Content adaptive complexity reduction scheme for quality/fidelity scalable HEVC," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1744–1748.

[57] Z.-Y. Chen and P.-C. Chang, "Rough mode cost–based fast intra coding for high-efficiency video coding," *Journal of Visual Communication and Image Representation*, vol. 43, pp. 77–88, 2017.

[58] R. A. Shah, M. N. Asghar, S. Abdullah, N. Kanwal, and M. Fleury, "SLEPX: An Efficient Lightweight Cipher for Visual Protection of Scalable HEVC Extension," *IEEE Access*, vol. 8, pp. 187 784–187 807, 2020.

[59] D. Wang, Y. Sun, C. Zhu, W. Li, and F. Dufaux, "Fast Depth and Inter Mode Prediction for Quality Scalable High Efficiency Video Coding," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 833–845, 2020.

[60] H. R. Tohidypour, H. Bashashati, M. T. Pourazad, and P. Nasiopoulos, "Fast Mode Assignment for Quality Scalable Extension of the High Efficiency Video Coding (HEVC) Standard: A Bayesian Approach," in *Proceedings of the 6th Balkan Conference in Informatics*, 2013, p. 61–65.

[61] L. Shen and G. Feng, "Content-Based Adaptive SHVC Mode Decision Algorithm," *IEEE Transactions on Multimedia*, vol. 21, no. 11, pp. 2714–2725, 2019.

[62] Y. Chih-Hsuan, L. Jie-Ru, C. Mei-Juan, Y. Chia-Hung, L. Cheng-An, and T. Kuang-Han, "Fast prediction for quality scalability of High Efficiency Video Coding Scalable Extension," *Journal of Visual Communication and Image Representation*, vol. 58, pp. 462–476, 2019.

[63] X. Li, M. Chen, Z. Qu, J. Xiao, and M. Gabbouj, "An Effective CU Size Decision Method for Quality Scalability in SHVC," *Multimedia Tools and Applications*, vol. 76, 2017.

[64] W. Chou-Chen, C. Yuan-Shing, and H. Ke-Nung, "Efficient Coding Tree Unit (CTU) Decision Method for Scalable High-Efficiency Video Coding (SHVC) Encoder," in *Recent Advances in Image and Video Coding.* Rijeka: IntechOpen, 2016, ch. 11.

[65] D. Wang, C. Zhu, Y. Sun, F. Dufaux, and Y. Huang, "Efficient Multi-Strategy Intra Prediction for Quality Scalable High Efficiency Video Coding," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 2063–2074, 2019.

[66] H. R. Tohidypour, M. T. Pourazad, and P. Nasiopoulos, "Probabilistic Approach for Predicting the Size of Coding Units in the Quad-Tree Structure of the Quality and Spatial Scalable HEVC," *IEEE Transactions on Multimedia*, vol. 18, no. 2, pp. 182–195, 2016.

[67] T. Hamid Reza, M. T. Pourazad, and P. Nasiopoulos, "Content adaptive complexity reduction scheme for quality/fidelity scalable HEVC," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1744–1748.

[68] T. Katayama, W. Shi, T. Song, and T. Shimamoto, "Early depth determination algorithm for enhancement layer intra coding of SHVC," in *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 3079–3082.

[69] D. Wang, C. Yuan, Y. Sun, J. Zhang, and H. Zhou, "Fast Mode and Depth Decision Algorithm for Intra Prediction of Quality SHVC," 2014, pp. 693–699.

[70] T. Wiegand, J.-R. Ohm, G. J. Sullivan, W.-J. Han, R. Joshi, T. K. Tan, and K. Ugur, "Special Section on the Joint Call for Proposals on High Efficiency Video Coding (HEVC) Standardization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1661–1666, 2010.

[71] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, 2012.

[72] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera, "HEVC Deblocking Filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1746–1754, 2012.

[73] C.-M. Fu, E. Alshina, A. Alshin, Y.-W. Huang, C.-Y. Chen, C.-Y. Tsai, C.-W. Hsu, S.-M. Lei, J.-H. Park, and W.-J. Han, "Sample Adaptive Offset in the HEVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1755–1764, 2012.

[74] R. Fan, Y. Zhang, and B. Li, "Motion Classification-Based Fast Motion Estimation for High-Efficiency Video Coding," *IEEE Transactions on Multimedia*, vol. 19, no. 5, pp. 893–907, 2017.

[75] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive Fractional-Pixel Motion Estimation Skipped Algorithm for Efficient HEVC Motion Estimation," vol. 14, no. 1, pp. 1551–6857, 2018.

[76] T. Koga, "Motion Compensated Inter-Frame Coding for Video Conferencing," 1981.

[77] K. Chun and J. Ra, "An improved block matching algorithm based on successive refinement of motion vector candidates," *Signal Processing: Image Communication*, vol. 6, no. 2, pp. 115–122, 1994.

[78] L.-W. Lee, J.-F. Wang, J.-Y. Lee, and J.-D. Shie, "Dynamic search-window adjustment and interlaced search for block-matching algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 1, pp. 85–87, 1993.

[79] R. Li, B. Zeng, and M. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[80] J. Jain and A. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Transactions on Communications*, vol. 29, no. 12, pp. 1799–1808, 1981.

[81] M.-J. Chen, L.-G. Chen, and T.-D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 5, pp. 504–509, 1994.

[82] F. Chen, P. Wen, Z. Peng, G. Jiang, M. Yu, and H. Chen, "Hierarchical complexity control algorithm for HEVC based on coding unit depth decision," *EURASIP Journal on Image and Video Processing*, vol. 96, 2018.

[83] R. Khemiri, N. Bahri, F. Belghith, F. E. Sayadi, M. Atri, and N. Masmoudi, "Fast motion estimation for HEVC video coding," in *2016 International Image Processing, Applications and Systems (IPAS)*, 2016, pp. 1–4.

[84] R. Saran, H. B. Srivastava, and A. Kumar, "Median predictor-based lossless video compression algorithm for IR image sequences," in *Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA 2007*, 2007, pp. 87–90.

[85] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

[86] C.-H. Cheung and L.-M. Po, "Novel cross-diamond-hexagonal search algorithms for fast block motion estimation," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 16–22, 2005.

[87] M. HEVC, " Software reference test model HM 16.5," `https://hevc.hhi.fraunhofer.de/`, [online; accessed 04-July-2019].

[88] G. Bjontegaard, "Calculation of average PSNR differences between RD curves," *ITU-T SG16/Q6 Document*, 2001.

[89] B. G., "Improvements of the BD-PSNR model," *ITU-T SG16/Q6 Document*, 2008.

[90] D. Liu, X. Liu, and Y. Li, "Fast CU Size Decisions for HEVC Intra Frame Coding Based on Support Vector Machines," in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2016, pp. 594–597.

[91] A. Hamosfakidis and Y. Paker, "A Novel Hexagonal Search Algorithm for Fast Block Matching Motion Estimation," *EURASIP Journal on Advances in Signal Processing*, 2002.

[92] M. Ismail, J. Ma, and D. Sim, "Full depth RQT after PU decision for fast encoding of HEVC," in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*, 2014, pp. 1–2.

[93] H.-S. Kim and R.-H. Park, "Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 130–138, 2016.

[94] H. L. Tan, C. C. Ko, and S. Rahardja, "Fast Coding Quad-Tree Decisions Using Prediction Residuals Statistics for High Efficiency Video Coding (HEVC)," *IEEE Transactions on Broadcasting*, vol. 62, no. 1, pp. 128–133, 2016.

[95] J. Leng, L. Sun, T. Ikenaga, and S. Sakaida, "Content Based Hierarchical Fast Coding Unit Decision Algorithm for HEVC," in *2011 International Conference on Multimedia and Signal Processing*, vol. 1, 2011, pp. 56–59.

[96] X. Shen, L. Yu, and J. Chen, "Fast coding unit size selection for HEVC based on Bayesian decision rule," in *2012 Picture Coding Symposium*, 2012, pp. 453–456.

[97] S. Cho and M. Kim, "Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 9, pp. 1555–1564, 2013.

[98] H.-M. Yoo and J.-W. Suh, "Fast coding unit decision algorithm based on inter and intra prediction unit termination for HEVC," in *2013 IEEE International Conference on Consumer Electronics (ICCE)*, 2013, pp. 300–301.

[99] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Fast HEVC Encoding Decisions Using Data Mining," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 4, pp. 660–673, 2015.

[100] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and Multi-Class Learning Based Low Complexity Optimization for HEVC Encoding," *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 547–561, 2017.

[101] H. K. Joy, M. R. Kounte, and A. K. Joy, "Deep Learning Approach in Intra - Prediction of High Efficiency Video Coding," in *2020 International Conference on*

Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), 2020, pp. 134–138.

[102] T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," 2017, pp. 1255–1260.

[103] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing Complexity of HEVC: A Deep Learning Approach," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044–5059, 2018.

[104] Z. Chen, J. Shi, and W. Li, " Learned Fast HEVC Intra Coding ," *IEEE Transactions on Image Processing*, vol. 29, pp. 5431–5446, 2020.

[105] J. Shi, C. Gao, and Z. Chen, "Asymmetric-Kernel CNN Based Fast CTU Partition for HEVC Intra Coding," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[106] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, "CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5088–5103, 2016.

[107] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, and L. Xu, "Machine Learning-Based Coding Unit Depth Decisions for Flexible Complexity Allocation in High Efficiency Video Coding," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2225–2238, 2015.

[108] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[109] W. Hamidouche, M. Farajallah, M. Raulet, O. Déforges, and S. El Assad, "Selective video encryption using chaotic system in the shvc extension," 04 2015.

[110] R. Bailleul, J. De Cock, and R. Van De Walle, "Fast mode decision for SNR scalability in SHVC digest of technical papers," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*, 2014, pp. 193–194.

[111] Model, " SHVC Reference Software," `https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/tags/SHM-12.1`, [online; accessed 04-July-2021].

[112] J. Kim and E.-S. Ryu, "Qos optimal real-time video streaming in distributed wireless image-sensing platforms," *Journal of Real-Time Image Processing*, vol. 13, 09 2017.

[113] Y. Ye and P. Andrivon, "The Scalable Extensions of HEVC for Ultra-High-Definition Video Delivery," *IEEE MultiMedia*, vol. 21, no. 3, pp. 58–64, 2014.

[114] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian, "Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 20–34, 2016.

[115] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[116] M. Xu, X. Deng, S. Li, and Z. Wang, "Region-of-Interest Based Conversational HEVC Coding with Hierarchical Perception Model of Face," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 3, pp. 475–489, 2014.

[117] S. Gokturk, C. Tomasi, B. Girod, and C. Beaulieu, "Medical image compression based on region of interest, with application to colon CT images," in *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 3, 2001, pp. 2453–2456.

[118] H. Yu, Z. Lin, and F. Pan, "Applications and improvement of H.264 in medical video compression," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 12, pp. 2707–2716, 2005.

[119] Y. Wu, P. Liu, Y. Gao, and K. Jia, "Medical ultrasound video coding with H.265/HEVC based on ROI extraction," *PLOS ONE*, vol. 11, 11 2016.

[120] S. Khire, S. Robertson, N. Jayant, E. A. Wood, M. E. Stachura, and T. Goksel, "Region-of-interest video coding for enabling surgical telementoring in low-bandwidth scenarios," in *MILCOM 2012 - 2012 IEEE Military Communications Conference*, 2012, pp. 1–6.

[121] D. Grois, E. Kaminsky, and O. Hadar, "ROI adaptive scalable video coding for limited bandwidth wireless networks," in *2010 IFIP Wireless Days*, 2010, pp. 1–5.

[122] T. Barsakar and V. Mankar, "A novel approach for medical video compression using kernel based meanshift ROI coding techniques," in *2016 Conference on Advances in Signal Processing (CASP)*, 2016, pp. 212–216.

[123] M. Ghafoor, A. Tariq, M. Bakr, Jibran, W. Ahmad, and T. Zia, "Perceptually Lossless Surgical Telementoring System Based on Non-Parametric Segmentation," *Journal of Medical Imaging and Health Informatics*, vol. 9, pp. 464–473, 03 2019.

[124] W. Xie, Z. Yao, E. Ji, H. Qiu, Z. Chen, H. Guo, J. Zhuang, Q. Jia, and M. Huang, "Artificial Intelligence–based Computed Tomography Processing Framework for Surgical Telementoring of Congenital Heart Disease," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 17, pp. 1–24, 10 2021.

[125] P. Liu, C. Li, C. Xiao, Z. Zeshu, J. Ma, J. Gao, P. Shao, I. Valerio, T. Pawlik, C. Ding, A. Yilmaz, and R. Xu, "A Wearable Augmented Reality Navigation System for Surgical Telementoring Based on Microsoft HoloLens," *Annals of Biomedical Engineering*, vol. 49, 06 2020.

[126] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[127] H. Wei, X. Zhou, W. Zhou, C. Yan, Z. Duan, and N. Shan, "Visual saliency based perceptual video coding in HEVC," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 2547–2550.

[128] L. Wang, P. C. Pedersen, D. M. Strong, B. Tulu, E. Agu, and R. Ignotz, "Smartphone-Based Wound Assessment System for Patients With Diabetes," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 2, pp. 477–488, 2015.

[129] A. Ramya, R.and Jenitta, "Foot injury detection using K-means clustering, mean shift segmentation algorithm," *Int J Adv Res Basic Eng Sci Technol (IJARBEST)*, vol. 3, no. 24, pp. 323–329, 2017.

[130] H. Wannous, S. Treuillet, and Y. Lucas, "Robust tissue classification for reproducible wound assessment in telemedicine environments," *J. Electronic Imaging*, vol. 19, p. 023002, 04 2010.

[131] C. Wang, X. Yan, M. Smith, K. Kochhar, M. Rubin, S. M. Warren, J. Wrobel, and H. Lee, "A unified framework for automatic wound segmentation and analysis with deep convolutional neural networks," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, pp. 2415–2418.

[132] M. Goyal, N. D. Reeves, A. K. Davison, S. Rajbhandari, J. Spragg, and M. H. Yap, "DFUNet: Convolutional Neural Networks for Diabetic Foot Ulcer Classification," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 728–739, 2020.

[133] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[134] T. McClellan, "Z-Plasty of Scar Contracture (Finger)," `https://youtu.be/wdseg3UvXrI`, [online; accessed 07-July-2021].

[135] McClellan, "The digital nerve was cut," `https://youtu.be/CY1HYIBrAwQ`, [online; accessed 05-July-2021].

[136] T. McClellan, "Flexor Digitorum Profundus (FDP) Finger Tendon Repair," `https://youtu.be/boMlEa3P43g`, [online; accessed 06-July-2021].

[137] McClellan, "Foreign Body (BB) Removal from Finger," `https://youtu.be/DWQ6WX3ImBU`, [online; accessed 07-July-2021].

[138] T. McClellan, "Ganglion Cyst: Flexor Tendon Sheath (Finger)," `https://youtu.be/hDZBE8tcctE`, [online; accessed 04-July-2021].

[139] McClellan, "Ganglion Cyst Volar Wrist," `https://youtu.be/ZgNJ8YDA7dY`, [online; accessed 04-July-2021].

[140] Vangelisti, "NuGrip Arthroplasty (Thumb Arthritis Joint Replacement Surgery)," https://youtu.be/YZgDQl5kWFs, [online; accessed 06-July-2021].

[141] T. McClellan, "Small Finger Extensor Tendon Saw Injury Cut Repair," https://youtu.be/3o7cgZsd3bs, [online; accessed 04-July-2021].

[142] McClellan, "Running Subcuticular Suture," https://youtu.be/CiW93U-3XcQ, [online; accessed 05-July-2021].

[143] A. Hassan, M. Ghafoor, A. Tariq, T. Zia, and W. Ahmad, "High Efficiency Video Coding (HEVC)–Based Surgical Telementoring System Using Shallow Convolutional Neural Network," *Journal of Digital Imaging*, vol. 32, 04 2019.

[144] A. Betka, N. Terki, A. Toumi, M. Hamiane, and A. Ourchani, "A new block matching algorithm based on stochastic fractal search," *Applied Intelligence*, vol. 49, no. 3, p. 1146–1160, 2019.

[145] L. Lin, I.-C. Wey, and J.-H. Ding, "Fast predictive motion estimation algorithm with adaptive search mode based on motion type classification," *Signal, Image and Video Processing*, vol. 10, pp. 171–180, 2016.

[146] M. Wu, X. Li, C. Liu, M. Liu, N. Zhao, J. Wang, X. Wan, Z. Rao, and L. Zhu, "Robust global motion estimation for video security based on improved k-means clustering," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 439–448, 2019.

[147] "A hybrid block-based motion estimation algorithm using jaya for video coding techniques," *Digital Signal Processing*, vol. 88, pp. 160–171, 2019.

[148] H. Amirpour, M. Ghanbari, A. Pinheiro, and M. Pereira, "Motion estimation with chessboard pattern prediction strategy," *Multimedia Tools and Applications*, vol. 78, pp. 1–20, 08 2019.

[149] E. Cuevas, "Block-matching algorithm based on harmony search optimization for motion estimation," *Applied Intelligence*, vol. 39, no. 1, p. 165–183, 2013.

[150] G. Senbagavalli and R. Manjunath, "Motion estimation using variable size block matching with cross square search pattern," *SN Applied Sciences*, vol. 2, 2020.

[151] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman, "Visual dynamics: Stochastic future generation via layered cross convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 41, no. 9, pp. 2236–2250, 2019.

[152] V. Vukotic, S. Pintea, C. Raymond, G. Gravier, and J. C. van Gemert, "One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network," *CoRR*, vol. abs/1702.04125, 2017.

[153] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *CoRR*, vol. abs/1511.05440, 2016.

[154] J. J. Hintz, "Generative adversarial reservoirs for natural video prediction," Ph.D. dissertation, 2016.

[155] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, "Learning to generate long-term future via hierarchical prediction," in *Proc. Int. Conf. Mach. Learn.* JMLR.org, 2017, p. 3560–3569.

[156] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. JMLR.org, 2015, p. 843–852.

[157] M. Oliu, J. Selva, and S. Escalera, "Folded recurrent neural networks for future video prediction," in *Computer Vision – ECCV 2018*, Cham, 2018, pp. 745–761.

[158] V. Michalski, R. Memisevic, and K. Konda, "Modeling deep temporal dependencies with recurrent "grammar cells"," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Cambridge, MA, USA: MIT Press, 2014, p. 1925–1933.

[159] E. Denton and V. Birodkar, "Unsupervised learning of disentangled representations from video," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2017, p. 4417–4426.

[160] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," vol. 34, 2020, pp. 12 993–13 000.

[161] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4473–4481.