3D-HEVC Fast Partionining Algorithm Based on MD-CNN

Nacir Omran, Imen Werda, Amna Maraoui, Rostom Kachouri, Hamdi Belgacem

3D-HEVC Fast Partionining Algorithm Based on MD-CNN

Nacir Omran
LEμE
University of Monastir
Monastir, Tunisia
E-mail: nacir.omran2@gmail.com

Imen Werda
LETI
University of Sfax
Sfax, Tunisia
E-mail: imen.phd@gmail.com

Amna Maraoui
LEμE
University of Monastir
Monastir, Tunisia
E-mail: maraouiamna@gmail.com

Rostom Kachouri
LIGM
Gustave Eiffel University, ESIEE Paris
Paris, France
E-mail: rostom.kachouri@esiee.fr

Hamdi Belgacem
LEμE
University of Monastir
Monastir, Tunisia
E-mail: belgacem.hamdi@gmail.com

***Abstract*—The introduction of the 3D-HEVC encoding standard has revolutionized the field of 3D and multi-view video by effectively synthesizing 3D video with adequate depth effects using depth map sequences. However, achieving this involves computationally intensive operations, such as the quad-tree partitioning of Intra Coding Units (CUs). Researchers used multiple approaches including heuristic, machine learning and deep learning to address this complexity. We opted to train a Multi-Deep Convolutional Neural Network (MD-CNN) model specifically for depth maps and integrated it into the 3D-HEVC encoder. This modified encoder uses the independent views as a reference to encode the dependent depth map views. This strategy reduces the complexity of the 3D-HEVC video encoder by 70.12% while slightly reducing video compression efficiency by 0.02% and a small Peak signal-to-noise ratio (PSNR) penalty of -0.80 dB.**

***Index Terms*—3D-HEVC, Video Encoding, Multi-Deep Convolutional Neural Network**

I. Introduction

The advent of 3D technology has revolutionized visual experiences in entertainment, even though special glasses are still required. Two primary methods for presenting 3D videos are the multiview video with depth (MVD) format by the Moving Picture Experts Group (MPEG) and the widely adopted auto-stereoscopic mode. MVD utilizes depth maps and 2D textures to generate virtual views using view synthesis and depth-image-based rendering (DIBR). To meet the increasing need for MVD coding technologies, the International Organization for Standardization (ISO) and the International Telecommunication Union (ITU) created the Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V). This team extended the HEVC standard to create three-dimensional high-efficiency video coding (3D-HEVC) [1], a compression technique for 3D videos with depth and multi-view texture data. In 3D-HEVC, texture and depth map videos are compressed by employing inter-view prediction to exploit correlations between different views and a tailored quadtree structure. It further improves compression efficiency by capitalizing on redundancy within the texture and depth map data through sophisticated inter-component coding techniques.

The fundamental framework of 3D-HEVC is illustrated in Figure 1, which shows that texture and depth map videos for a predetermined camera position are represented by a view identifier (”Viewid”). This identifier also dictates the coding sequence. The view marked with ”Viewid” 0 is called the independent or base view and is encoded independently using a standard HEVC encoder [2]. Other views, known as dependent views, utilize additional inter-view prediction tools beyond those defined in the HEVC standard for rate-distortion optimization (RDO) [3].

The 3D-HEVC architecture, shown in Figure 1, identifies every texture and depth map video using a Viewid. The main view (Viewid 0) is independently encoded with a standard HEVC encoder [2], while dependent views are encoded using inter-view prediction and rate distortion (RD). The quad-tree coding block partitioning structure of HEVC enables the use of small and large prediction blocks. In the HEVC reference encoder (HM), images are structured into tree blocks, which are then divided into coding units (CUs). These CUs can range in size from the full dimensions of the tree block down to the minimum prediction block size of 8x8 pixels. HEVC supports four levels of coding unit (CU) depths: 64x64, 32x32, 16x16, and 8x8, each subdivided into two or four prediction units (PUs). Intra-coded CUs use PU sizes such as 2Nx2N and NxN, while inter-coded CUs offer a broader range of PU sizes, leading to different inter-frame prediction modes, including SKIP mode, Merge mode, Intra 2Nx2N, Intra NxN, Inter 2Nx2N, Inter 2NxN, Inter 2NxnU, Inter 2NxnD, Inter nLx2N, Inter nRx2N, and Inter NxN (for the smallest CU). SKIP and Merge modes utilize previously coded data for motion vectors, eliminating the need to encode additional motion data. The mode decision process of the 3D-HEVC encoder's coding unit selection module assesses all possible prediction modes and CU depth levels to choose the best one. This is done

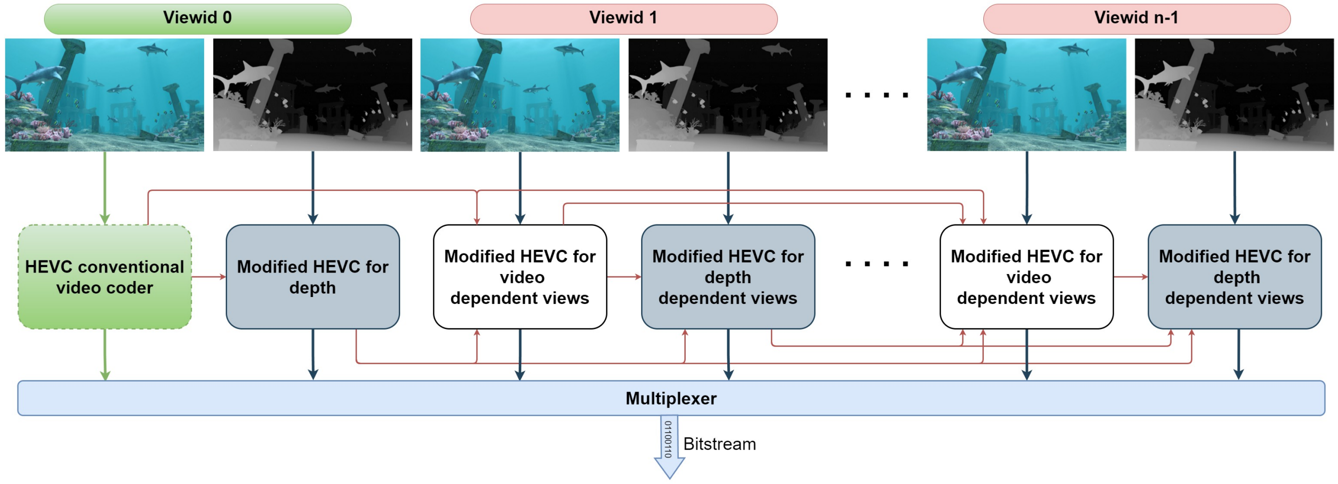
Viewid 0
Viewid 1
Viewid n-1
. . . .
HEVC conventional video coder
Modified HEVC for depth
Modified HEVC for video dependent views
Modified HEVC for depth dependent views
. . . .
Modified HEVC for video dependent views
Modified HEVC for depth dependent views
Multiplexer
0100110
Bitstream

Fig. 1. 3D-HEVC view encoding framework.

by establishing a rate-distortion cost (RD cost) list where the values are compared to determine the combination with the lowest RD cost. The selection process of the optimal prediction mode and CU depth level might require a significant amount of computational resources, especially for real-time encoding of high-definition videos (4K and 8K). This complexity is compounded by the inter-view prediction, which involves variable-sized motion estimation (ME) and disparity estimation (DE) for every dependent view, making it the most demanding part of a 3D-HEVC system [3]. Consequently, this complexity stands as an obstacle to practical implementation and real-time applications. Depth maps often contain high spatial resolution and have varying levels of fine detail and texture, especially in complex scenes. This demands an extremely flexible and adaptive partitioning scheme to effectively capture the details with minimal data. The 3D-HEVC encoder involves a hierarchical structure where each CU can be recursively divided into smaller units. This critical process adaptively decides the size of each CU based on local content characteristics, which involves complex decision algorithms and adds to the computational burden. The recursive nature of CU division and the need to evaluate multiple partitioning schemes to find the optimal balance make it computationally demanding. This is often the most time-consuming part of the depth map coding process [3], This poses a significant obstacle to the widespread adoption of 3D-HEVC in real-world applications. By addressing the complexity of CU division, depth map coding can be made more efficient, leading to better performance in various real-time applications like streaming or interactive environments. Facing the new challenges, we are motivated to accelerate depth map coding module in order to solve the above problems. Ongoing research focuses on improving the algorithms for CU division, including heuristic-based methods like [7] and [8], machine learning-based methods like [9] and [10], and deep learning-based methods [11] that can predict optimal partitions more efficiently.

Our contribution focuses on mitigating the aforementioned complexity by modifying the standard 3D-HEVC encoder and integrating a multi-level convolutional neural network (CNN) for coding tree unit (CTU) partitioning and using the independent depth view as a reference and for encoding the dependent views.

The remainder of this paper is organized as follows, starting with describing the intricacies of the coding structure of 3D-HEVC with diving deeper into the intra-frame coding process and the CTU partitioning process using quad-tree structure. Next, we discuss the proposed method for CTU partitioning based on deep learning where we explain the proposed model MD-CNN architecture and establish a database. Then we move to the experimental results where analyse our findings concerning compression, quality and time reduction, and compare these results to other works. Finally, we end with the conclusion where we summarise our work.

II. 3D-HEVC ENCODING FRAMEWORK

A. *3D-HEVC ENCODING FRAMEWORK*

3D-HEVC (High-Efficiency Video Coding) extends the H.265/HEVC standard to support stereoscopic and multiview video coding. While it retains the basic HEVC framework, including the use of coding tree units (CTUs) as the fundamental coding blocks, 3D-HEVC incorporates unique features to handle multiple views and depth information. Each CTU in 3D-HEVC is composed of multiple slices, with each slice representing a different view. These slices are encoded independently, which facilitates efficient parallel processing.

Besides the previously mentioned modes (intra and inter), 3D-HEVC incorporates an inter-view prediction mode, enabling one view to be predicted from another, a feature particularly beneficial for stereoscopic and multiview videos where views are often similar. To accommodate depth information, a new and modified mode for depth coding was added to 3D-HEVC, encoding separate depth maps for each view. These depth maps enable advanced compression techniques like view synthesis and depth-based view interpolation.

B. 3D-HEVC intra-frame encoding framework

The intra-frame coding framework in 3D-HEVC parallels that of HEVC, starting with the segmentation of the frame into coding tree units (CTUs), which are 64x64 pixel blocks. Each CTU is further divided into coding units (CUs) of varying sizes and shapes, which are fundamental for intra-prediction and transform coding. In 3D-HEVC, intra-prediction is performed using neighboring pixels within the same slice or view, and also supports inter-view prediction through the use of corresponding CUs from different views. The residual signal, derived by subtracting the predicted signal from the original, is transformed using the discrete cosine transform (DCT) to convert spatial domain coefficients into the frequency domain. These coefficients are then quantized and encoded through methods such as context-adaptive binary arithmetic coding (CABAC), introducing a degree of lossy compression. The encoded data is subsequently stored or transmitted. During the decoding process, the data undergoes entropy decoding, inverse quantization, and inverse transformation to reconstruct the residual signal, which is then combined with the predicted signal to recreate the original frame.

C. CTU partitioning process using quad-tree structure

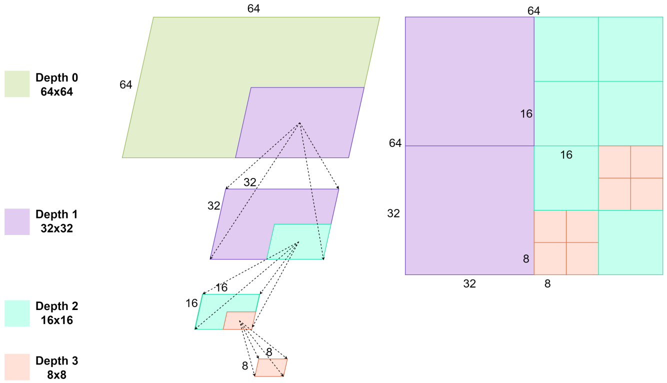

Fig. 2. Partitionning structure of 3D-HEVC.

The CTU division in 3D-HEVC is based on the original encoder HEVC, which means that every frame is segmented into several non-overlapping Coding Tree Units (CTUs). It is shown in figure 2 that each CTU can consist of a single CU with a size of 64x64 or be subdivided into smaller CUs of sizes 32x32, 16x16 and 8x8. These sizes are referred to as depth as they describe how deep is the subdivision from depth 0 being the whole CTU down to depth 3 being the smallest at 8x8. It's important to note that the optimal partition structure of a CTU is determined through iterative computation. The quadtree partitioning process of a CTU involves both a top-down RD-cost calculation and a bottom-up RD-cost comparison.

As illustrated in Figure 2 the RD-cost for each coding unit at the current depth is calculated during the top-down calculation process in the following order: depth=0 depth=1 depth=2 depth=3. The bottom-up comparison procedure is then carried out. In the event that RD-cost(depth=n) RD-cost(depth=n+1) n=0 1 2 further division of the coding unit at depth=n is not required. Based on statistics more than 90% of the total complexity of depth map coding is accounted for by the intricacy of the coding unit division. Full traversal mode encoding for a 64×64 CTU necessitates at least 2623 RD-cost calculations, 1935 Sum of Absolute Transformed Difference (SATD) cost calculations and 85 CU calculations. Thus it is essential to simplify the depth map coding unit division in 3D-HEVC.

III. Proposed Fast CTU partitioning using multi deep convolutional neural network(MD-CNN)

A. Proposed model architecture

In our approach, we developed a partitioning prediction model using a Multi-Deep Convolutional Neural Network MD-CNN to enhance the efficiency of Coding Tree Unit (CTU) partitioning for depth maps within the 3D-HEVC framework. According to [13], MD-CNN is multiple convolutional neural networks with different inputs as needed for the application and concatenated in one single model. It was applied in [14] for drawing a partitioning map for HEVC. As seen in Figure 3, the first convolutional layers are responsible for extracting the features for depth 0 64×64 partitioning if not the next depth of convolutional layers is used to extract features for the smaller partitioning at depth 1 and depth 2. If the model determines that any of the 3 partitioning sizes (64×64, 32×32 and 16×16) are not suitable it automatically assigns depth 3 (8×8). This model eliminates the need for time-consuming recursive rate-distortion cost (RD-Cost) calculations by directly predicting split decisions.

The inputs to our model are independent viewpoints that match CTUs in the 64x64 depth map. Each channel starts by loading the respective block size as an array of pixel values. The initial layer of each channel extracts low-level features such as edges and orientations using 2×2 convolutions with a stride of 4. To capture higher-level features, we apply additional convolutional layers with 2×2 kernels, utilizing 24 filters in the second layer and 32 filters in the third layer.

The vectorized features from these convolutional layers are subsequently passed through three fully connected layers. The first two hidden layers generate feature vectors, and the final output layer produces three sets of outputs: Output1, Output2, and Output3, containing 1, 4, and 16 binary elements, respectively. We use the Rectified Linear Unit (ReLU) activation function in the convolutional layers to introduce non-linearity, which enhances the model's ability to learn complex patterns found in the depth map. This activation function is described in the following equation:

$$C_m(\text{CTU}_n) = \begin{cases} C_0(\text{CTU}_n) = \text{CTU}_n, & m = 0 \\ \text{ReLU}(W_m \times C_{m-1}(\text{CTU}_n) + B_m), & 1 \leq m \leq M \end{cases} \quad (1)$$

Here, C_m represents the convolutional layers following the preprocessing module, where M signifies the total number

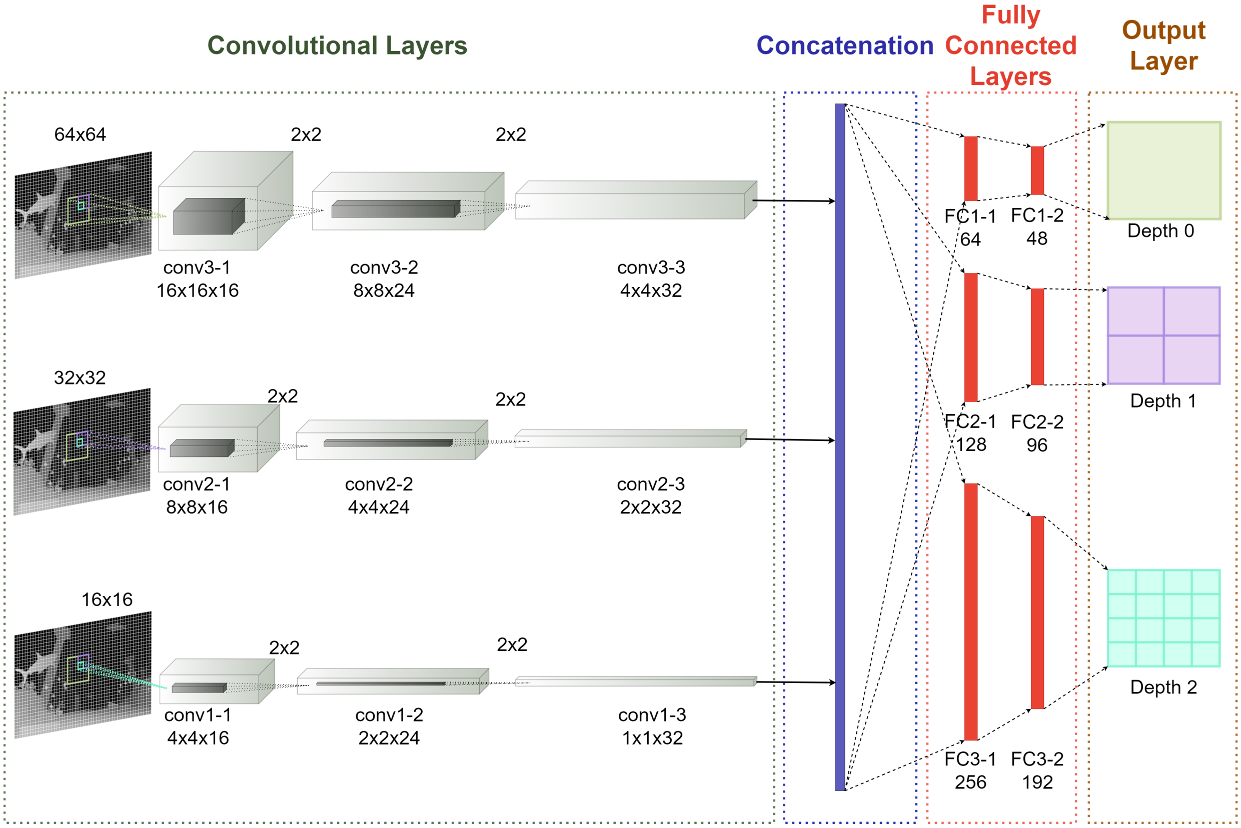

Fig. 3. Multi deep convolutional neural network architecture used for 3D-HEVC CTU partitioning.

of convolutional layers, m represents the current layer being processed, and n represents the current CTU being processed. Wm represents the weight matrix, and Bm represents the bias. Taking into consideration the model's output as binary classification (indicating division or not of the CU with 0 and 1), the sigmoid activation function is employed in the final layer of the proposed model.

During MD-CNN training, the three channels are trained simultaneously as part of an end-to-end model. The collaborative MD-CNN model utilizes the binary cross-entropy loss function, defined as

$$L = -\frac{1}{N}\sum_{n=1}^{N}\left(l_n \times \ln Y_n + (1 - l_n) \times \ln\left(1 - Y_n\right)\right) \tag{2}$$

In this context, $l_n \in \{0, 1\}$ indicates if the current CU is divided, N is the total number of samples, n denotes the current sample being processed, and Y_n represents the model's output.

By integrating reference view partitioning data, our Multi-Deep Convolutional Neural Network (MD-CNN) method optimizes the prediction of Coding Tree Unit (CTU) partitioning for depth maps within the 3D-HEVC framework. This integration is pivotal as it allows the MD-CNN to capitalize on spatial correlations between different views seen in Figure 1, as the independent views are merely the same as the dependent view but with a few degrees of a shift in perspective to the left and right corresponding to the camera setup. These correlations enable the network to make informed decisions about how best to partition CTUs, thereby reducing redundancies in partitioning and reducing complexity by accurately predicting partitioning based on learned patterns from reference views.

B. Database establishment

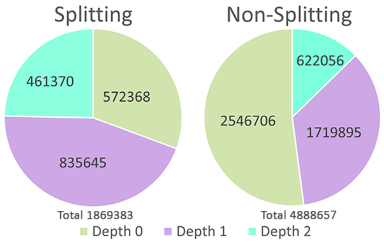

Fig. 4. Distribution of dataset samples: Split and Non-Split samples for each depth.

The efficiency of our MD-CNN model in predicting CTU partitioning is influenced by the selection of data. The number and type of dataset items can significantly influence its performance. We choose to utilize high-resolution texture images [12] instead of evaluating sequence segments, as done in previous studies, due to the absence of a depth image database. Our judgment is further supported by the utilization of the HM16.18 [5] encoder throughout the encoding process. The HTM16.3 [6] encoder, also known as the texture 2D encoder, is referred to as the CTU partitioning process in the 3D-HEVC encoder. The HEVC reference program HM16.18 is utilized to encode a set of 2500 pictures with different resolutions (4928×3264, 2880×1920, 1536×1024, 1792×1024, and 768×512) at different quantization parameters (34, 39, 42, 48) to obtain the matching 64x64 split decision map. This is done to guarantee optimal precision and performance. The dataset is divided into three folders: training (2125 shots), validation (125 images), and test (250 images). Each folder has all CU sizes, along with the corresponding binary labels for various QPs. The photographs used for training, validation, and testing are mutually exclusive and do not have any overlap. 6758040 tagged samples are gathered in total, as indicated in Figure 4. In order to assure the stability of our database, the fraction of split CU samples is almost equal to that of non-split samples.

IV. Experimental results

TABLE I
Sequences and its related information grouped by class

Sequences	Frame Rate	Three viewpoints	Resolution	Number Frames
Balloons	30	1-3-5	1024x768	300
Newspaper	30	2-4-6	1024x768	300
Shark	30	1-5-9	1920x1088	300
Ghost town fly	25	9-5-1	1920x1088	250
Poznan hall	25	7-6-5	1920x1088	200
Undo dancer	25	1-5-9	1920x1088	250

To assess the performance of the proposed methods, experiments were carried out using the 3D-HEVC reference software version HTM16.3 [6], which includes depth and texture encoders derived from HEVC reference software version HM16.18 [5]. To simplify the HEVC intra-coding process, we utilized the configuration file baseCfg_3view+depth_AllIntra.cfg. Testing was conducted on seven video sequences specified in Table I.

For evaluating coding efficiency, video quality, and complexity reduction, we employed metrics such as the Bitrate-Distortion Bound Rate (BDBR), Bjontegaard Delta Peak Signal-to-Noise Ratio (BDPSNR), and time reduction (TR) as defined in eq. 3.

$$TR = \frac{T_{Proposed} - T_{Original}}{T_{Original}} \times 100 \tag{3}$$

TABLE II
Experimental environment

Operating system	CPU	GPU	RAM	Framwork
Windows 10 Professional 64-bit	Intel Core i7-8700 @ 3.20 GHz	Nvidia GeForce GTX 1050 Ti @1291 MHz	16 GB	TensorFlow

The tests were conducted on the setup seen in II. The CNNs were implemented using the TensorFlow [4] framework and trained with three classifiers across four QP values (34, 39, 42, 48). For benchmarking purposes, we utilized well-established video sequences listed in Table I, featuring synchronized texture and depth images from three cameras (one main view and two auxiliary views). We processed 100 frames from each sequence with both the standard 3D-HEVC reference software version HTM16.3 and our custom-integrated CNN model, applying four different QP pairings for encoding texture and depth (25-34, 30-39, 35-42, 40-48).

According to the observed results, there is a significant reduction in encoding time with an average of -70.12% across all QP pairs compared to the reference encoding time. This reduction indicates the efficiency of the proposed method in decreasing computational complexity.

Analyzing the correlation between QP pairs and time reduction, we observe that QP 25-34 averaged a time reduction of -67.40%, while QP 40-45 averaged -71.13%. This slight variation suggests that higher QP pairs tend to achieve more time reduction, which may be due to the CU size distribution favouring larger CUs at higher QPs.

Bitrate fluctuations are minor across different sequences. The worst-case scenario shows a bitrate decrease of -1.66% for the Ghost Town Fly sequence at QP 25-34, while the best-case scenario indicates a bitrate increase of 1.58% for the Balloons sequence at QP 35-42. Overall, the average bitrate change across all sequences and QP pairs is just 0.02%, suggesting a negligible impact on compression efficiency.

When looking at the relationship between QP pairs and bitrate, a slight improvement is evident with higher QP pairs. The average bitrate change for QP 25-34 is -0.80%, whereas for QP 40-45, there is a minor increase of 0.07%. This trend, combined with the observed time reduction, suggests that the proposed model is more effective at reducing computational complexity and maintaining bitrate at higher QPs.

Regarding video quality, there is negligible degradation across the sequences. The average Y-PSNR change for QP 25-34 and QP 30-39 is -1.55 dB and -0.78 dB, respectively. For the other QP pairs, the average Y-PSNR change is -0.65 dB and -0.33 dB. This indicates that the proposed method maintains video quality reasonably despite the significant reductions in encoding time.

When comparing the performance metrics (BDBR% and TR%) of different methods across various sequences (Balloons, Newspaper, Shark, Ghost town fly, Poznan hall, Undo dancer), several observations can be made. The proposed work consistently shows higher BDBR values and greater time reduction compared to other methods across most sequences. For instance, the Balloons sequence achieves a bitrate of 0.91% and a time reduction of -72.33%, which are higher than, respectively, those of Chen2021 (0.19%, -45.29%) and Lin 2021 (0.12%, -34.31%). Similar trends are observed in other sequences such as Newspaper, where the proposed work reports a BDBR of 1.11% and a TR% of -73.81%. Looking at the average performance across all sequences, the proposed work maintains an average BDBR of 0.02%, which indicates a lesser compression penalty compared to Chen2021 (0.32%), Chang2019 (0.12%), Lin 2021 (0.15%), but fall behind compared to our previous work Omran 2023 (-0.15%). However, its average TR% is notably lower at -70.12%, compared to Chen2021 (-56.08%), Chang2019 (-59.21%), Lin 2021 (-45.35%), and Omran 2023 (-48.46%). Overall, these results suggest that the proposed work demonstrates competitive

TABLE III

BITRATE, PSNR, AND TIME REDUCTION RESULTS FOR EACH QP PAIR USING OUR CUSTOM 3D-HEVC INTEGRATED WITH MD-CNN

	QP pairs														
	QP 25 -34			QP 30-39			QP 35-42			QP 40-45			Average		
Sequences	Δ Bitrate %	ΔY-PSNR%	Δ TR%	Δ Bitrate %	Δ Y-PSNR%	Δ TR%	Δ Bitrate %	Δ Y-PSNR%	Δ TR%	Δ Bitrate %	Δ Y-PSNR%	Δ TR%	Δ Bitrate %	ΔY-PSNR%	ΔTR%
Balloons	0.18	-0.52	-68.85	1.15	0.15	-72.07	1.58	0.02	-73.57	0.72	-0.47	-74.83	0.91	-0.21	-72.33
Newspaper	-0.01	-0.72	-71.58	1.15	0.54	-73.61	1.94	-0.07	-74.29	1.36	-0.48	-75.77	1.11	-0.18	-73.81
Shark	-0.98	-3.31	-65.31	-2.71	-1.99	-67.02	-1.27	-1.13	-66.30	-0.45	0.19	-69.56	-1.35	-1.56	-67.05
Ghost town fly	-1.66	0.01	-69.23	-0.84	0.49	-69.01	-0.32	-0.05	-66.50	-0.08	-0.03	-68.88	-0.72	0.10	-68.41
Poznan hall	-1.19	-0.60	-62.36	-1.22	-0.29	-66.27	-1.08	-0.23	-70.60	-0.73	-0.14	-71.68	1.11	-0.18	-73.81
Undo dancer	-1.15	-4.13	-67.06	-1.16	-3.60	-64.01	-0.91	-2.43	-64.11	-0.40	-1.04	-66.07	-0.90	-2.80	-65.31
Average	-0.80	-1.55	-67.40	-0.60	-0.78	-68.66	-0.01	-0.65	-69.23	0.07	-0.33	-71.13	0.02	-0.80	-70.12

TABLE IV

COMPARISON OF BITRATE AND TIME REDUCTION WITH OTHER WORKS

	Chen2021		Chang2019		Lin 2021		Omran 2023		Proposed work	
Sequences	BDBR %	TR%	BDBR %	TR%	BDBR %	TR%	BDBR %	TR%	BDBR %	TR%
Balloons	0.19	-45.29	0.10	-46.56	0.12	-34.31	0.54	-47.63	0.91	-72.33
Newspaper	0.53	-42.48	0.02	-36.77	0.35	-34.36	0.71	-52.00	1.11	-73.81
Shark	0.22	-58.92	0.03	-64.04	0.06	-45.92	-1.87	-46.52	-1.35	-67.05
Ghost town fly	0.25	-56.16	0.07	-65.50	0.02	-46.62	-0.46	-46.95	-0.72	-68.41
Poznan hall	0.71	-72.25	0.40	-79.12	0.36	-59.69	0.71	-52.00	1.11	-73.81
Undo dancer	0.28	-66.47	0.15	-61.38	0.06	-52.67	-0.56	-45.64	-0.90	-65.31
Average	0.32	-56.08	0.12	-59.21	0.15	-45.35	-0.15	-48.46	0.02	-70.12

bitrate performance and considerable improvement in time reduction of -70.12%. However, we fall behind our previous work from -0.09 dB to -0.8 dB in terms of BD-PSNR which indicates a a slight degradation in video quality. This trade-off between encoding speed and compression quality should be taken into consideration and improved in later work to close the gap between the two metrics.

V. Conclusion

This work presents a novel method that reduces the major complexity related to the quadtree division of depth map intra-coding units in 3D-HEVC. Using reference view partitioning data, it suggests an effective method for depth map intra-coding unit division based on MD-CNN. This integration is essential because it allows the MD-CNN to make use of spatial correlations between various perspectives, which improves its capacity to efficiently forecast Coding Tree Unit (CTU) partitioning. By directly predicting CTU division topologies, the technique seeks to do away with the necessity for computationally demanding rate-distortion cost computations. A 3D-CNN can be trained on a large dataset to teach it how to automatically examine input CTUs and anticipate the best division structures. The MD-CNN, when included in the 3D-HEVC encoder, slightly reduces coding efficiency by 0.02% and reduces coding complexity by an average of 70.12% with only a penalty of -0.8dB to DB-PSNR.

References

[1] G. Balota, M. Saldanha, G. Sanchez, B. Zatt, M. Porto, and L. Agostini, ”Overview and quality analysis in 3D-HEVC emergent video coding standard,” in 2014 IEEE 5th Latin American Symposium on Circuits and Systems, 2014, pp. 1–4.

[2] G. J. Sullivan *et al.*, ”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] G. Sanchez, L. Agostini, and C. Marcon, *Algorithms for Efficient and Fast 3D-HEVC Depth Map Encoding*. Springer, 2020.

[4] M. Abadi et al., ”TensorFlow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[5] ITU/ISO/IEC, ”HEVC HM reference software,” 2017. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/HM/-/tree/HM-16.18?ref_type=tags. [Accessed: 28-June-2024].

[6] ITU/ISO/IEC, ”3D-HEVC HTM reference software,” 2017. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/branches/HTM-16.3-fixes/. [Accessed: 28-June-2024].

[7] J.-R. Lin et al., ”Visual perception based algorithm for fast depth intra coding of 3D-HEVC,” *IEEE Transactions on Multimedia*, vol. 24, pp. 1707–1720, 2021.

[8] M.-J. Chen *et al.*, ”Fast 3D-HEVC depth intra coding based on boundary continuity,” IEEE Access, vol. 9, pp. 79588–79599, 2021.

[9] C.-H. Fu *et al.*, ”Fast depth intra coding based on decision tree in 3D-HEVC,” IEEE Access, vol. 7, pp. 173138–173147, 2019.

[10] M. Saldanh *et al.* ”Fast 3D-HEVC depth map encoding using machine learning.” IEEE Transactions on Circuits and Systems for Video Technology 30.3 (2019): 850-861.

[11] N. Omran *et al.*, ”Fast partition algorithm in depth map intra coding unit based on multi-deep convolution neural network,” Journal of Real-Time Image Processing, vol. 21, no. 1, p. 23, 2024.

[12] D.T. Dang-Nguyen *et al.* ”Raise: A raw images dataset for digital image forensics.” Proceedings of the 6th ACM multimedia systems conference. 2015.

[13] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang and Z. Guan, ”Reducing Complexity of HEVC: A Deep Learning Approach,” in IEEE Transactions on Image Processing, vol. 27, no. 10, pp. 5044-5059, Oct. 2018, doi: 10.1109/TIP.2018.2847035.

[14] Amna, M., Imen, W., Soulef, B. *et al.* Machine Learning-Based approaches to reduce HEVC intra coding unit partition decision complexity. Multimed Tools Appl 81, 2777–2802 (2022). https://doi.org/10.1007/s11042-021-11678-2