Received January 10, 2022, accepted January 24, 2022, date of publication January 28, 2022, date of current version February 7, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3147502

Early Termination of CU Partition Based on Boosting Neural Network for 3D-HEVC Inter-Coding

SIHAM BAKKOURI[1] **AND ABDERRAHMANE ELYOUSFI**[2]
[1]Computer Systems and Vision Laboratory, Faculty of Sciences, Ibn-Zohr University, Agadir 80000, Morocco
[2]Department of Computer Science, National Engineering School of Applied Sciences, Ibn-Zohr University, Agadir 80000, Morocco
Corresponding author: Siham Bakkouri (siham.bakkouri@gmail.com)

ABSTRACT As an extension of the High Efficiency Video Coding (HEVC) standard, the 3D-HEVC needs to encode multiple texture videos and depth maps components. In the 3D-HEVC inter-coding test model, a large variety of Coding Unit (CU) sizes are adopted to select the one with the lowest Rate-Distortion (RD) cost as the best CU size. This technique provides the highest achievable coding efficiency, but it brings a huge computational complexity which limits 3D-HEVC from practical applications. In this paper, early termination of CU encoding is proposed to reduce the complexity caused by the CU size splitting process. The proposed algorithm is based on CU homogeneity and a boosting neural network clustering algorithm. The algorithm contained three main steps. The first step is for the extraction of various features from the original encoder. Then, the selection of the features, which had a high correlation with CU partition using a machine learning algorithm. In the second step, a boosting neural network model is used for training the selected features to derive the threshold values for our proposed algorithm. In the final step, an efficient early termination of CU splitting is released for texture videos and depth maps based on the extracted thresholds from the training model. The experimental results show that the proposed algorithm reduces a significant encoding time, while the loss in coding efficiency is negligible.

INDEX TERMS 3D-HEVC, feature selection, machine learning, inter prediction, boosting neural network.

I. INTRODUCTION

With the fast recent development of 3D multimedia and displaying technologies, the three-Dimensional (3D) video systems have evolved due to their real-world visual experience that goes beyond two-Dimensional (2D) videos [1]. In the Multi-View plus Depth (MVD) systems, which consists of multiple texture video and associated depth maps, a small number of captured texture video and its corresponding depth map are coded and the resulting bitstream packets are multiplexed into a 3D video bitstream [2], [3].

3D-HEVC is an extension of the HEVC standard [4], [5], which was developed by the Joint Collaborative Team on 3D Video (JCT-3V). 3D-HEVC introduces many techniques and feature tools to further improve coding efficiency. When considering the inter-coding process, 3D-HEVC adopts a hierarchical coding structure which is one of the most powerful tools to improve the coding efficiency of the 3D-HEVC encoder [6], [7]. However, this hierarchical coding structure is based on the quad-tree structure of the Coding Unit (CU). As shown in Figure 1, the largest CU is called Coding Tree Unit (CTU) and each CTU is divided into four CUs. Besides, the CU is split recursively into four equally CU sizes. These CU sizes are varying from 64×64 to 8×8, corresponding to depth levels 0 to 3, respectively [8]. The 3D-HEVC encoder evaluates all CU sizes using the Lagrange multiplier to catch the one with the least Rate Distortion (RD) cost as the best CU split [9]. The RD-cost function is presented as follows:

$$RD - cost = SSE_{luma} + \omega_{chroma} \times SSE_{chroma} + \lambda \times B \quad (1)$$

where SSE_{luma} and SSE_{chroma} are the average difference between the current CU and the matching CUs in the luma component and chroma component, respectively, ω_{chroma} is the chroma weighing factor, λ is the Lagrange multiplier, B is bit cost to be considered for 3D-HEVC mode decision. This flexible coding structure of 3D-HEVC contributes significant improvement in coding gain. However, it brings a dramatic increase in encoding complexity because the encoding

The associate editor coordinating the review of this manuscript and approving it for publication was Chaker Larabi.

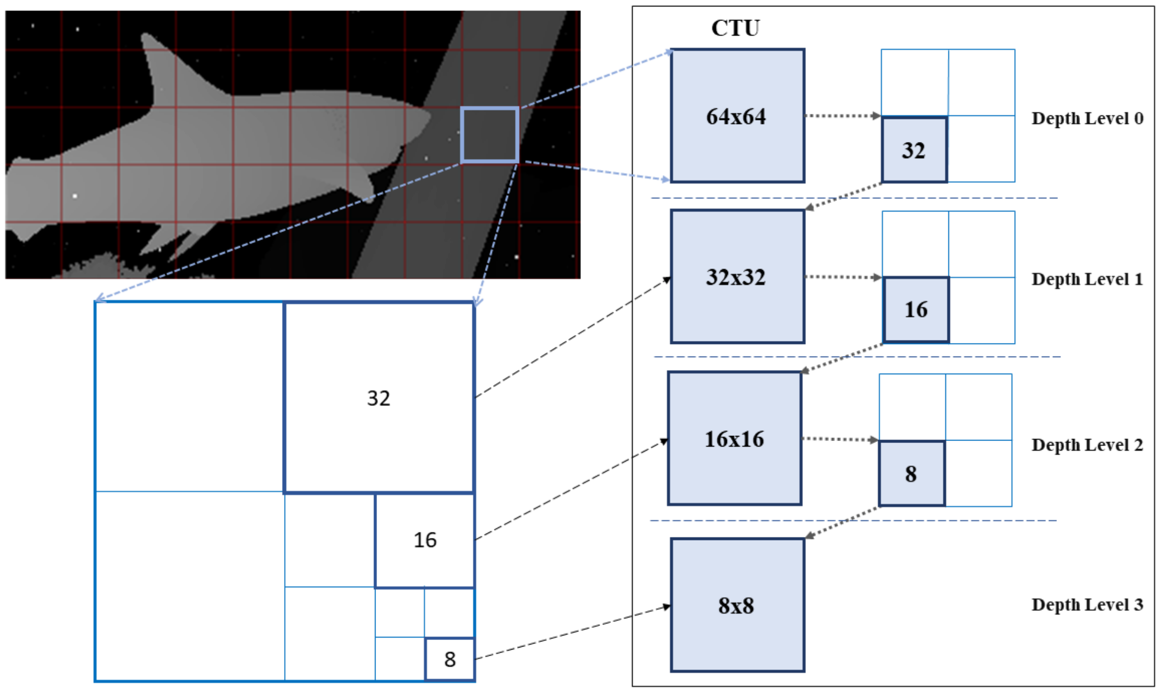


FIGURE 1. **CU split process in 3D-HEVC inter-coding.**

process of 3D-HEVC needs to explore every single CU from 64 × 64 to 8 × 8 size where the best CU partition must be decided for all possible CU sizes. Therefore, it is required to develop an early termination of CU encoding algorithm to reduce the computational complexity of the 3D-HEVC encoder.

Several fast inter coding approaches have been proposed to reduce the complexity of 3D-HEVC encoder [10]–[15]. In our previous work [10], we proposed a fast CU size decision algorithm based on a machine learning algorithm. Thus, The structure tensor is adopted as an extracted feature to build a binary classification model. This algorithm is used to extract the adaptive splitting values for depth maps and texture CUs. An early termination of CU size and fast merge decision algorithms are proposed in [11] for 3D-HEVC inter-coding. The inter-view correlation is used as a priori information to select the optimal prediction unit (PU) and CU sizes. A fast inter prediction algorithm is proposed in [12] using depth map segmentation. The proposed work is based on the CTU classification into uniform and complex CTUs by dividing the depth map into three parts (foreground, middle-ground, and background) for an early termination of depth map CU encoding. In [13], the authors proposed a fast inter mode decision algorithm to reduce the complexity of the 3D-HEVC coding. The proposed algorithm is based on two mean steps, firstly, an early skip mode decision is built based on the texture correlation of adjacent dependent and base views. Then, the symmetric and asymmetric motion partition modes are checked by selectively skipping according to the texture feature of the coding CU. In [14], a fast inter and intra mode decision algorithm is proposed based on the correlation between the depth and the texture videos, and edge classification. While the texture videos and their associated depth map represent the same scene at the same time instant, the CU sizes and prediction modes rarely used in the corresponding texture video are skipped from the associated depth map views to reduce the 3D-HEVC coding complexity. In [15], the authors proposed a low complexity mode decision algorithm for inter and intra prediction using the depth map changes characteristics to skip the unnecessary depth modes for both inter and intra prediction.

Although, all the existing are efficient to reduce the computational complexity of 3D-HEVC inter-coding. Therefore, there is still some room left for further complexity reduction. In our previous work [10], we proposed a fast CU size decision algorithm using tensor features as homogeneity determination and a machine learning binary classification model. In this paper, we propose an amelioration of the previous work based on an early termination of the CU encoding algorithm by investigating various features in training sets to reduce the complexity of texture videos and depth maps. The CU size decision algorithm is modeled as a data classification problem which is applied to predict whether the current CU split or not to the smaller sizes. The classification is efficiently solved using the AdaBoosting Neural Network (AB-NN) algorithm. The AB-NN algorithm is used in this investigation due to the high accuracy for this kind of problem, given that it can derive the adaptive thresholds for texture and depth maps CUs. The experimental results demonstrate that the proposed algorithm can significantly reduce the computational complexity of 3D-HEVC with negligible coding performance degradation compared to the original encoder.

The remainder of the paper is organized as follows. Section II provides the statistical analysis of CU partitions in 3D-HEVC encoder. Section III introduces the boosting neural network algorithm. The 3D-HEVC features analysis is presented in Section IV. The proposed algorithm is described in Section V. Performance evaluations of the proposed algorithm are shown in Section VI, and the conclusion of this work is provided in Section VII.

II. MOTIVATION AND STATISTICAL ANALYSIS

3D-HEVC adopts a hierarchical coding structure, which is one of the most powerful tools to improve the coding efficiency of the 3D-HEVC encoder [6], [7]. In the joint model of 3D-HEVC, a complex RD optimization process is performed for all the possible CU sizes to find one with the minimum RD cost and determine the best coding size for a CU. Thus, the small CU sizes are likely to be chosen for coding the complex region and small CU sizes are more suitable for coding the homogenous region. However, the depth maps are mainly characterized by a large homogeneous region, where they have a higher probability to be coded using larger CU sizes. Therefore, a proposed algorithm based on the CU homogeneity could skip the RD Cost time-consuming process computed on the CU split process, and then reduce significantly the computational complexity of the 3D-HEVC coding process.

In order to well understand the correlation between the CU complexity and the CU size decision in the 3D-HEVC encoder, we encoded eight experimental videos sequences using the HTM-16.3, Random Access (RA) configurations [16]. The experimental videos are recommended by Common Test Conditions (CTC) [17] using four Quantization Parameters(QPs). The experiments covered the 3D video sequences: "Balloons", "Kendo" and "Newspaper" with a resolution of 1024×768, and "GT_Fly", "Poznan_Hall2", "Poznan_Street", "Undo_Dancer" and "Shark" with a resolution of 1920×1088. The four pairs of quantization parameters (QP-pairs) are used to encode texture and depth maps (QP-texture, QP-depth), which are (25, 34), (30, 39), (35, 42) and (40, 45). Table 1 and Table 2 show the distribution of CU sizes for texture videos and depth maps, respectively, according to the four QP-pairs. For texture views, it can be seen from Table 1 that The probability of a CU to be coded with large sizes is very high compared to the small CU sizes for all sequences and all QP-pairs, it is more than 85% on average. While the total percentage of CUs with small CU sizes is less than 15% on average. Furthermore, the probability of choosing the size 64×64 is depended on the complexity of the sequence. For "Shark" which is characterized by complex regions, the proportion of 64×64 is approximately 64% for small QP, and for "Poznan_Hall2" which is characterized by homogenous regions, the proportion of 64×64 is more than 84% in small QP. For the depth maps, the probability of choosing CU size 64×64 is more than 82% for all sequences and 96.16% on average. This occurs because the depth map is characterized by sharp edges and large homogeneous regions differing from typical texture video contents. Therefore, the probability of choosing small sizes is less than 5% on average. It can also be seen that the variety of quantization parameters QP affects the CU size distribution for both texture videos and depth maps. Thus, the QP describes the compression rate by impacting the image quality. High QPs generate more homogeneous areas in the coded image that are efficiently encoded using larger CU sizes. However, with low QPs, the predicted images tend to preserve several details, requiring smaller CU sizes to manage the encoding efficiency. Therefore, if we can decide the CU size and skip the CU splitting process, the coding time will be saved and the computational complexity of the 3D-HEVC encoder can be reduced.

III. BOOSTING NEURAL NETWORK ALGORITHM

A boosting is an approach for developing the performance of learning algorithms. The boosting algorithm is one of the most powerful learning techniques introduced during the past decade. The motivation for the boosting algorithm is to produce a scheme that combines many "weak" classifiers [18] such as decision trees and neural networks to achieve a powerful classifier. The most promising boosting algorithm is "Adaptive Boosting", namely, AdaBoost, which is introduced by Freund and Schapire [19]. AdaBoost has been applied with large success to manifold benchmark machine learning problems using mainly decision trees as base classifiers [10], [20]. Besides, there is recent evidence that AdaBoost may very well overfit if we combine several hundred thousand classifiers. It also seems that the performance of AdaBoost degrades a lot in the presence of significant amounts of noise [21], [22]. However, to make the AdaBoost model more efficient, many works have already proposed AdaBoost with the neural network algorithm as a weak learner instead of the decision tree in the traditional AdaBoost models [23]–[25]. In [23], the authors reported that the Adaboosting neural network is significantly better than boosted decision trees in terms of accuracy. Although the neural network is reported to be outperformed by some statistical methods in most cases, attempts to improve it have never been stopped [26], [27].

A neural network (or an Artificial Neural Network) [28] is a learning processing model that is inspired by the way of biological nervous systems, such as the brain to treat information. The key element of this model is the novel structure of the information processing system. It is constituted of a large number of highly interconnected processing elements namely, neurons, working in harmony to solve specific problems. The most popular type of neural network is composed of three layers of units: input layers, hidden layers, and output layers as shown in Figure 2. The input layer is connected to hidden layer, which is also linked to the output layer. The input layer activity is the raw information that is introduced into the network. The activity of each hidden unit is given

TABLE 1. CU size distribution of texture for 3D-HEVC (RA configuration).

Sequences	QP	64× 64	32×32	16×16	8× 8
Balloons	25	7.,92%	17.10%	4.96%	1.02%
	30	86.24%	10.55%	2.79%	0.43%
	35	90.75%	7.55%	1.50%	0.21%
	40	94.01%	5.02%	0.88%	0.09%
Kendo	25	72.54%	19.86%	6.48%	1.12%
	30	80.31%	14.81%	4.28%	0.60%
	35	85.68%	11.43%	2.60%	0.30%
	40	89.84%	8.44%	1.55%	0.17%
Newspaper	25	73.93%	17.41%	6.93%	1.74%
	30	85.32%	10.50%	3.48%	0.69%
	35	91.30%	6.69%	1.73%	0.28%
	40	95.03%	4.03%	0.80%	0.13%
GT_Fly	25	78.90%	14.77%	4.98%	1.34%
	30	87.98%	9.01%	2.43%	0.58%
	35	93.76%	4.87%	1.16%	0.20%
	40	96.85%	2.54%	0.55%	0.06%
Poznan_Hall2	25	87.46%	10.35%	1.96%	0.23%
	30	93.91%	5.17%	0.85%	0.06%
	35	96.52%	3.16%	0.30%	0.02%
	40	98.30%	1.53%	0.17%	0.01%
Poznan_Street	25	74.71%	16.54%	7.45%	1.30%
	30	85.52%	10.63%	3.36%	0.49%
	35	91.68%	6.59%	1.62%	0.12%
	40	95.23%	4.00%	0.74%	0.03%
Undo_Dancer	25	68.14%	21.64%	8.35%	1.87%
	30	80.26%	14.56%	4.33%	0.85%
	35	88.04%	9.40%	2.30%	0.27%
	40	92.84%	6.00%	1.06%	0.10%
Shark	25	64.73%	24.11%	9.12%	2.04%
	30	77.05%	17.63%	4.49%	0.83%
	35	85.54%	12.05%	2.13%	0.29%
	40	91.93%	7.02%	0.97%	0.08%
Average		85.98%	10.47%	3.01%	0.55%

by the activities of the input units and the weights (W_{ij}) over the connections between the input and the hidden units. Therefore, the transfer and the activation functions translate the input signals to output signals. The threshold value of our model is extracted from the activation function, in which the output is set at one of two levels, according to the fact that the total input is greater than or less than some threshold value. The performance of the output units is affected by the activity of the hidden units and the weights between the hidden and output units.

In this paper, we consider the neural network to be the base classifier of Adaboost model, namely, AdaBoosting Neural Network(AB-NN). We are given a training data set $S = \{(x_i, y_i), \ldots, (x_N, y_N)\}$, which x_i the input features and $y_i = \{-1,1\}$ corresponding to the output. In the AB-NN model, each sample in S is assigned an equal weight of 1/N, which means that each sample has the same opportunity to be selected at the first step. Generating T neural network classifiers for the AdaBoost model need T rounds of training neural network with T different training sample groups S_t (t $= 1, 2, \ldots,$ T). In round t, the function to determine the weight of sample i is denoted by D_t(i). In each round after the construction of the classifier AB-NN which provides a function h_t to map x to $\{-1,1\}$, the value of $D_t(i)$ is adjusted in terms of how they are classified by the classifier AB-NN and the training sample group S_{t+1} is then generated in terms of D_t on S with sample replacement.

TABLE 2. CU size distribution of depth maps for 3D-HEVC (RA configuration).

Sequences	QP	64× 64	32×32	16×16	8× 8
Balloons	34	89.98%	8.08%	1.73%	0.21%
	39	95.84%	3.47%	0.63%	0.06%
	42	97.96%	1.71%	0.31%	0.02%
	45	98.89%	1.01%	0.09%	0.01%
Kendo	34	86.92%	10.74%	2.10%	0.23%
	39	94.39%	4.79%	0.81%	0.02%
	42	97.49%	2.22%	0.27%	0.02%
	45	99.00%	0.88%	0.12%	0.00%
Newspaper	34	87.19%	9.31%	3.01%	0.49%
	39	94.50%	4.26%	1.09%	0.15%
	42	97.57%	1.86%	0.51%	0.06%
	45	98.90%	0.93%	0.15%	0.02%
GT_Fly	34	96.75%	2.61%	0.52%	0.11%
	39	99.30%	0.58%	0.10%	0.02%
	42	99.85%	0.14%	0.01%	0.00%
	45	99.97%	0.03%	0.00%	0.00%
Poznan_Hall2	34	97.83%	1.99%	0.18%	0.00%
	39	99.25%	0.68%	0.06%	0.00%
	42	99.73%	0.25%	0.02%	0.00%
	45	99.94%	0.05%	0.01%	0.00%
Poznan_Street	34	91.84%	6.25%	1.72%	0.20%
	39	96.59%	2.80%	0.58%	0.03%
	42	98.56%	1.26%	0.16%	0.01%
	45	99.51%	0.46%	0.03%	0.00%
Undo_Dancer	34	88.57%	8.64%	2.35%	0.44%
	39	94.31%	4.58%	0.98%	0.14%
	42	97.16%	2.37%	0.42%	0.05%
	45	98.88%	1.01%	0.10%	0.01%
Shark	34	82.47%	13.21%	3.62%	0.70%
	39	93.26%	5.51%	1.05%	0.18%
	42	97.30%	2.29%	0.37%	0.04%
	45	99.04%	0.85%	0.11%	0.01%
Average		95.90%	3.28%	0.72%	0.10%

The algorithm maintains a weight distribution D_t(i) over the data points. The weights are updated in each iteration. The goal of the base learner is to minimize the weighted error as follows:

$$\epsilon_t = \frac{1}{2}\sum_{i=1}^{N} D_t(i)\ [1-\theta_t(x_i)) \times y_i] \tag{2}$$

where $\theta_t(x_i)$ is a node in the output layer which indicate the threshold value that a CU can split or no into next depth levels. in this end, the output function h(x) can be defined as follows:

$$h_t(x) = \begin{cases} +1 & \text{if } x_i \leq \theta_t(x_i); \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

The final decision function for AB-NN algorithm is defined in Eq. 4, which βt is obtained by Eq. 5.

$$H_T(x) = \sum_{t=1}^{T}\left(log(\frac{1}{\beta_t})\right) h_t(x) \tag{4}$$

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t} \tag{5}$$

The details and the pseudo-code of the AB-NN model is described in Algorithm 1. Since, 0 <ϵ_t <0.5, so 0<β_t <1, the weights of the correctly classified samples with idea output is reduced by βt, and weights of those misclassified samples will have no change.

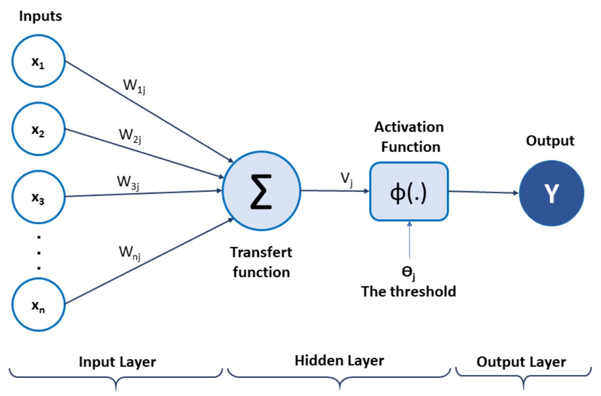


FIGURE 2. **Artificial neural network architecture.**

IV. 3D-HEVC CODING FEATURES ANALYSIS

A. FIRST AND SECOND ORDER STATISTICS FEATURES EXTRACTION

The precision of the CU size decision in a classification task is highly dependent on the feature space used to train the model. In most of the machine learning algorithms adopted in CU size decision of 3D-video coding, the features extracted from a CU are always statistic criteria, such as variance, structure tensor and gradient [7], [10], [29], [30]. In this paper, we are training two types of features measure. the first order and the second order features. The first order features which based on central moments [31], the texture measures are statistics calculated from an individual pixel and do not consider pixel neighbor relationships. In the second order or Gray Level Co-occurrence Matrix (GLCM) features, we consider the relationship between neighbors [32], [33].

1) FIRST ORDER FEATURES

First order features are based on statistical characteristics calculated from each CU. For a pixel I(i, j) in a luminance domain, the Mean m_1 and Central Moments μ_k for each CU are computed as follows:

$$m_1 = \frac{1}{N \times N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(i,j) \tag{6}$$

$$\mu_k = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I(i,j) - m_1)^k \tag{7}$$

where k = 2, 3 and 4, and N corresponds to the number of pixels for each CU size.

In this study, we are using the most frequently moments, which are the variance σ^2, the skewness and the kurtosis based on the central moments μ_2, μ_3, and μ_4 respectively. These features are called the normalized k-central moment and they are calculated as follows:

$$\sigma^2 = \frac{1}{N \times N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mu_2 \tag{8}$$

Algorithm 1 Pseudo-Code of AB-NN Algorithm

```
    Input: S, a set of samples for training with size N
T, the number of rounds to construct the AB-NN model
Settings for training the base neural network classifier
    Output: The classifier $H^T(x)$
    Init $D_1$(i)= 1/N, i = 1, ..., N

Require: Set max iterations $T$
  for $t \leftarrow 1$ to $T$ do
    Train neural network with respect to weight function $D_t$(i)
    Get weak hypothesis $h_t$
    Compute the weighted error $\epsilon_t$ using Eq.(2)
    if $\epsilon_t > 0.5$ then
      Set $D_t$(i)= 1/N, i = 1, ..., N
    end if
    while $\epsilon_t \leq 0.5$ do
      if $\epsilon_t = 0.5$ then
        Set T=t
        break
      else
        Compute the coefficient $\beta_t$ using Eq.(5)
        Update the weight function $D_t$(i):
        $D_{t+1}(\text{i}) = D_{t+1}(\text{i}) \times \beta_t^{\frac{1}{2}[1+\theta_t \times y_i]}$
        Normalize the weight function $D_{t+1}(i)$:
        $D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_{i=1}^{N} D_{t+1}(i)}$
      end if
    end while
  end for
  Calculate the classifier
  $H_T(x) = \sum_{t=1}^{T} \left( log(\frac{1}{\beta_t}) \right) h_t(x)$
```

$$Skewness = \frac{1}{N \times N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{\mu_3}{\sigma^3} \tag{9}$$

$$Kurtosis = \frac{1}{N \times N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{\mu_4}{\sigma^4} \tag{10}$$

2) SECOND ORDER FEATURES

The features generated from the first order provide information related to the gray-level distribution of the region. However, they are not given any information about the relative positions of the various gray levels within the region. These features will not be able to measure whether all low-value gray levels are positioned together, or they are interchanged with the high-value gray levels, especially depth maps CUs. For an analyzed image, the texture information is specified by the matrix of joint probability with two pixels separated by a distance of d along direction θ with gray levels i and j. We suppose the current CU is a matrix (donated as R) with N_x columns and N_y rows, G is the corresponding GLCM. Each pixel has eight nearest-neighbors connected to it, horizontal

0°, vertical 90°, right 45° and left-diagonal 135° directions, and their four contrary directions, as illustrated in Figure 3. It can be seen from the Figure 3 that Pixels 2 and 6 are 0° nearest pixels to pixel X, pixels 1 and 5 are 45° nearest neighbors, pixels 4 and 8 are 90° nearest neighbors, and pixels 3 and 7 are 135° nearest neighbors to pixel X. In this work, we calculate the GLCM for each pixel in the direction 0° only.

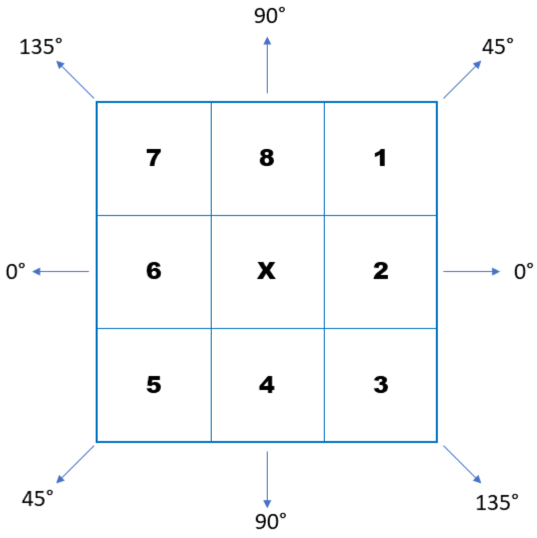


FIGURE 3. Eight nearest-neighbourhood scheme for obtaining co-occurrence matrix.

The value of matrix G in row *i*, column *j* can be calculated as follows:

$$G(i, j, d, \theta) = \#\{[(k, l), (m, n)] \in D \quad |R(k, l) = i, R(m, n) = j, d, \theta\} \tag{11}$$

where # represents the number of elements in the set, (k, l) and (m, n) are the coordinates in the pixel matrix R, and D = $(N_x \times N_y) \times (N_x \times N_y)$.

The GLCM for the direction 0° is calculated as follows:

$$\begin{aligned} &G(i, j, d, 0°) \\ &\quad = \#\{[(k, l), (m, n)] \in D \\ &\qquad |R(k, l) = i, R(m, n) = j, |l - n| = d, k - m = 0\} \end{aligned} \tag{12}$$

where R(m, n) and R(k, l) are the gray values in the matrix R. Finally, the GLCM matrix G can be represented as Eq.13, shown at the bottom of the page, where W is the gray level. To reduce the computational overhead, the GLCM matrix is calculated for $\theta = 0°$ and d = 1 and also the W is set to 8. Although, the luminance value range of the current CU is from 0 to 255, so each pixel need to be divided by 32.

In this study, four GLCMs features; Homogeneity, Contrast, Entropy and Angular Second Moment(ASM), were implemented. The mathematical description of these features are given in [32] and they are calculated as follows:

$$Homogeneity = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{G(i, j)}{1 + (i - j)^2} \tag{14}$$

$$Contrast = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i - j)^2 G(i, j) \tag{15}$$

$$Entropy = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} -G(i, j) \ln(G(i, j)) \tag{16}$$

$$ASM = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} G^2(i, j) \tag{17}$$

where G(i, j) is the GLCM matrix when d = 1 and $\theta = 0°$. Homogeneity is a measure that takes high values for low-contrast images, Contrast is a measure of local level variations which takes high values for image of high contrast, Entropy is a measure of randomness and takes low values for smooth images and ASM is a feature that measures the smoothness of the image. The less smooth the region is, the more uniformly distributed G(i, j) and the lower will be the value of ASM. All these features together provide high discriminative power to describe the complexity of a region.

B. 3D-HEVC FEATURES SELECTION

In this work, we are interested in reducing the encoding complexity of inter-prediction frames. To train our model, we encode five training video sequences with different resolutions (Non-CTC sequences). These video sequences are "Akko&Kayo" and "Rena" with a resolution of 640 × 448. "Pantomime", "Dog" and "Champagne_tower" with a resolution of 1220 × 960. The training dataset is collected from the first 100 frames of each sequence. We extract the first and second features for each CU and the splitting decision information. The splitting decision takes "0" when the CU is coded with the current size while taking the value "1" if the CU splits to the smaller sizes. We extracted these features for texture and depth maps separately for the four QP-pairs: (25, 34), (30, 39), (35, 42) and (40, 45), in which the RA configuration was considered. Although, the training

$$G = \begin{pmatrix} G(0, 0, 1, 0°) & G(0, 1, 1, 0°) & \cdots & G(0, W - 1, 1, 0°) \\ G(1, 0, 1, 0°) & G(1, 1, 1, 0°) & \cdots & G(1, W - 1, 1, 0°) \\ \vdots & \vdots & \ddots & \vdots \\ G(W - 1, 0, 1, 0°) & G(W - 1, 1, 1, 0°) & \cdots & G(W - 1, W - 1, 1, 0°) \end{pmatrix} \tag{13}$$

TABLE 3. Information gain for first and second features.

Features	Texture CU Sizes			Depth Map CU Sizes		
	64× 64	32×32	16×16	64× 64	32×32	16×16
Variance	0.03	0.22	0.18	0.02	0.16	0.13
Skewness	0.12	0.12	0.13	0.07	0.14	0.12
Kurtosis	0.01	0.13	0.14	0.00	0.11	0.12
Homogeneity	0.25	0.13	0.15	0.19	0.15	0.18
Contrast	0.24	0.14	0.14	0.27	0.17	0.18
Entropy	0.21	0.13	0.13	0.22	0.13	0.14
ASM	0.14	0.13	0.13	0.23	0.13	0.14

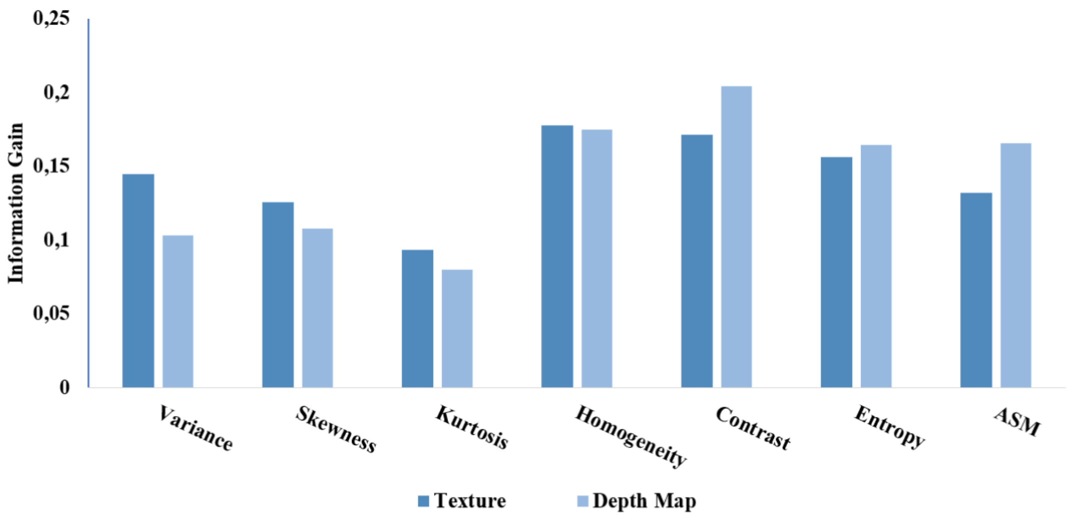


FIGURE 4. Average of IG for first and GLCM features.

dataset is composed of seven features and the splitting flag information.

To assess how each feature contributes to the CU partitioning decision, the Information Gain (IG) [34], [35] is used for this investigation. Among all evaluated features, some of them were selected, considering the correlation of each feature with the CU split decision. The algorithm applied, which is based on the Information Gain (IG), defines the most relevant features for dealing with the split decision. IG refers to the difference between the entropy of all data set and the entropy of the subset of the evaluated attribute. The Waikato Environment for Knowledge Analysis (WEKA) [36], version 3.8.5, was used to calculate the IG for the selected features. WEKA generated three datasets for texture and three datasets for depth maps. Each dataset is corresponding to a CU size, 64×64, 32×32 and 16×16, and it is composed of the seven training features and the splitting information flag. Considering the IG of the analyzed attributes associated with the CU split decision. Table 3 shows the IG of all features for CU sizes 64×64, 32×32 and 16×16. Figure 4 shows the average of IG for the tree CU sizes for texture and depth maps. It can be seen from Table 3 and Figure 4, that in texture CUs, the Homogeneity is ranked the first and the Contrast the second. Unless, in depth map CUs, the Contrast is ranked the first and the Homogeneity the second. To this end, the two GLCM features, Homogeneity and Contrast are selected for the AB-NN model.

V. THE EARLY TERMINATION OF CU ENCODING ALGORITHM

In this work, we proposed an early termination of CU encoding for texture videos and depth maps, in which this proposed approach can reduce the complexity of 3D-HEVC inter-coding. Our proposed method is based on the selected GLCM features in Section IV, and the AB-NN for training model which is done off-line. Thus, the training model is composed of three datasets for texture and three datasets for depth maps. Each dataset is corresponding to CU sizes 64×64, 32×32 and 16×16, and it is composed of the selected features Homogeneity and Contrast calculated by Eqs.(14) and (15), respectively, and the splitting decision information. The AB-NN algorithm is used to extract the threshold vector θ_j for each dataset corresponding the selected features as described in Algorithm (1).

Figure 5 shows the methodology for design and evaluation flow of our proposed algorithm. The first step is the extraction of features from the 3D-HEVC encoder using the training sequences. In the next step, the extracted features are filtering using WEKA and IG to select the ones with a high correlation with the CU partition. The selected features are using to build a training model using the AB-NN model and extract the threshold values for our proposed algorithm. Finally, the evaluation flows step is for performing our proposed algorithm using the extracted thresholds and compared the results with HTM 16.3 [16].

To simplify the use of the selected features for our proposed algorithm, we calculated in Eq.18, ω_{CU}, the distance between the selected features.

$$\omega_{CU} = \sqrt{H_{CU}^2 + C_{CU}^2} \tag{18}$$

where H_{CU} is the Homogeneity of each CU computed using Eq. 14, and H_{CU} is the Contrast of a CU calculated using Eq. 15. Therefore, the threshold value of the proposed algorithm is calculated in the same way using the extracted thresholds by AB-NN. The early termination of CU splitting (T_{enc}) is described as follows:

$$T_{enc} = \begin{cases} \text{Unsplit} & \text{if } \omega_{CU} \leq Th_S^V \\ \text{Split} & Otherwise \end{cases} \tag{19}$$

where Th_S^V is the threshold value, in which S parameter takes the values 0, 1, and 2 according to CU sizes, 64×64, 32×32, and 16×16, respectively. The parameter V takes T for texture CUs and D for depth map CUs. The value of the threshold Th_S^V, which is extracted from the AB-NN model explained in Section III, changes depending on the current view (texture or depth map), CU size, and the Quantization Parameters (QPs). If ω_{CU} smaller than Th_S^V, then, the current CU can be terminated early and coded with current size. Other than that, if ω_{CU} is larger than the selected Th_S^V, the current CU splits to the smaller sizes. The proposed overall algorithm is given in Algorithm 2.

It can be seen from Algorithm 2 that the *isDepth* flag is used for checking the texture and depth map CUs, and Th_0, Th_1, and Th_2 are the threshold values for CU sizes, 64×64, 32×32, and 16×16, respectively. Firstly, the GLCM filter is applied for each pixel, then, the checking process for texture and depth map CUs is started, in which, each incoming depth map CU, Th_0, Th_1, and Th_2 take the depth map thresholds Th_0^D, Th_1^D, and Th_2^D, respectively. And for each texture CU, Th_0, Th_1, and Th_2 take the texture thresholds Th_0^T, Th_1^T, and Th_2^T, respectively. Then, the splitting process is started and the ω_{CU} is calculated using Eq.(18) depending on the CU sizes. ω_0, ω_1, and ω_2 are the ω_{CU} according to CU sizes, 64×64, 32×32, and 16×16, respectively. Finally, an early termination of CU size is established to judge whether if the current CU must be or not be split into smaller sizes.

VI. EXPERIMENTAL RESULTS

To verify the efficiency of the early termination of the CU encoding algorithm, the proposed algorithm has been implemented on the recent 3D-HEVC reference software HTM-16.3 [16]. The coding experiments were defined under the Common Test Conditions (CTC) [17] required

Algorithm 2 Early Termination of CU Encoding Algorithm

```
Input: texture and depth map CTU
Compute GLCM filter for each pixel
if IsDepth then
    $Th_0 \leftarrow Th_0^D$;
    $Th_1 \leftarrow Th_1^D$;
    $Th_2 \leftarrow Th_2^D$;
else
    $Th_0 \leftarrow Th_0^T$;
    $Th_1 \leftarrow Th_1^T$;
    $Th_2 \leftarrow Th_2^T$;
end if
Calculate $Homogeneity_0$ and $Contrast_0$ for CU size $64 \times 64$

Calculate the distance $\omega_0$ using Eq. 18
if ($\omega_0 < Th_0$) then
    No split process
else
    Split CU to the quadtree $32 \times 32$
    for ( $i \leftarrow 0; i < 4; i \leftarrow i + 1$ ) do
        Calculate $Homogeneity_1$ and $Contrast_1$ for CU size $32 \times 32$
        Calculate $\omega_1$ using Eq. 18
        if ($\omega_1(i) < Th_1$) then
            No split process
            Split CU to the quadtree $16 \times 16$
            for ( $i \leftarrow 0; i < 4; i \leftarrow i + 1$ ) do
                Calculate $Homogeneity_2$ and $Contrast_2$ for CU size $16 \times 16$
                Calculate $\omega_2$ using Eq. 18
                if ($\omega_2(i) < Th_2$) then
                    No split process
                else
                    Split CU to the quadtree $8 \times 8$
                end if
            end for
        end if
    end for
end if
```

TABLE 4. Test sequences information.

Test Sequences	Resolution	Frames	3-View Input
Balloons	1024 × 768	300	1-3-5
Kendo	1024 × 768	300	1-3-5
Newspaper	1024 × 768	300	2-4-6
GT_Fly	1920 × 1088	250	9-5-1
Poznan_Hall2	1920 × 1088	300	7-6-5
Poznan_Street	1920 × 1088	250	5-4-3
Undo_Dancer	1920 × 1088	250	1-5-9
Shark	1920 × 1088	300	1-5-9

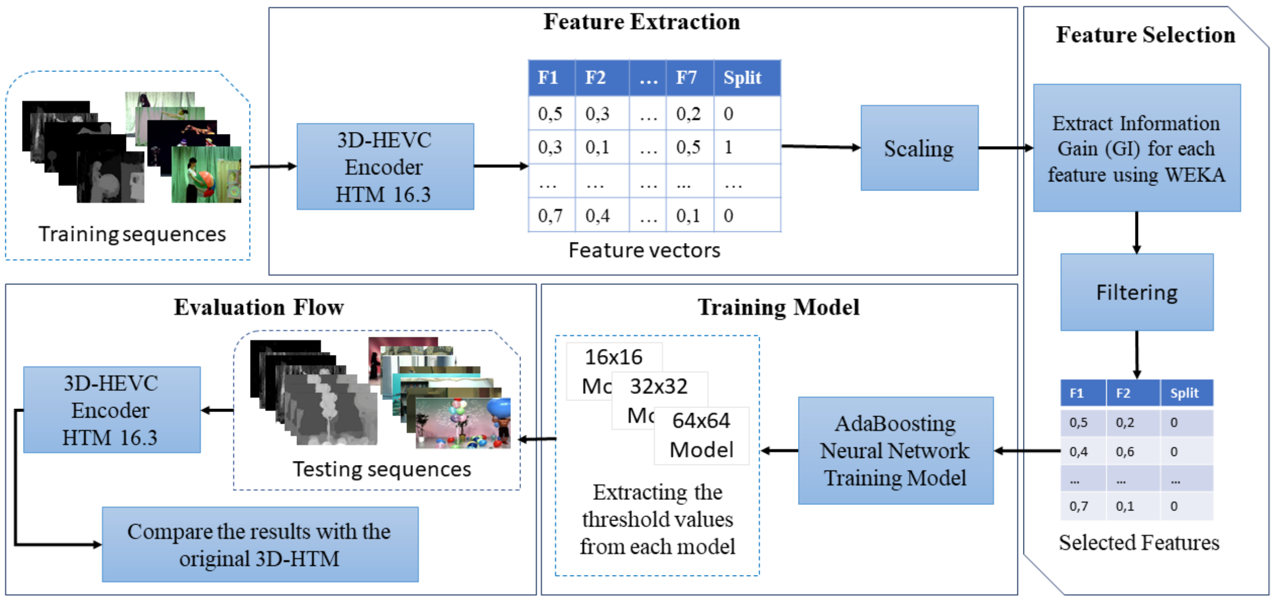


FIGURE 5. Methodology for design and evaluation of the proposed algorithm.

TABLE 5. Experimental results of the proposed algorithm compared with the original 3D-HEVC.

Sequences	Texture video		Synthesized views		TS (%)
	BD-BR (%)	BD-PSNR(dB)	BD-BR(%)	BD-PSNR (dB)	
Balloons	0.25	-0.02	0.19	-0.01	40.11
Kendo	0.31	-0.03	0.21	-0.02	39.36
NewsPaper	0.23	-0.01	0.23	-0.01	39.93
GT_Fly	0.29	-0.02	0.12	-0.01	41.23
Poznan_Hall2	0.32	-0.01	0.14	-0.01	41.48
Poznan_Street	0.27	-0.02	0.23	-0.02	40.29
Undo_Dancer	0.39	-0.02	0.28	-0.02	38.29
Shark	0.42	-0.03	0.32	-0.03	41.33
1024 × 768	0.26	-0.02	0.21	-0.01	39.80
1920 × 1088	0.34	-0.02	0.22	-0.01	40.52
Average	0.31	-0.02	0.22	-0.01	40.25

by JCT-3V using eight video test sequences presented as follows: "Kendo", "Balloons" and "Newspaper" with a resolution of 1024 × 768. "GT_Fly", "Poznan_Hall2", "Poznan_Street", "Undo_Dancer" and "Shark" with a resolution of 1920 × 1088. The 3-view case in CTC was used. In this case, for each sequence, 3 texture cameras and 3 associated depth maps were encoded using the random access configuration. Table 5 recapitulates the details of the CTC test sequences. The test platform is an Intel(R) Xeon(R) *CPU* E3-1225 v5 @ 3.30 GHz with 8 GB RAM and a Microsoft VS C++ 2015 compiler.

Table 5 presents the experimental results of the early termination of the CU encoding algorithm for the texture and depth maps in the 3D-HEVC encoder. We evaluate our proposed algorithm in respect to the Bjontegaard delta-rate (BD-BR, BD-PSNR) [37], [38] considering the quality of the synthesized views using the (VSRS) algorithm provided by JCT-3V [39]. TS represents the Time Savings of the entire encoder (texture and depth maps) defined as follows:

$$TS(\%) = \frac{ET_{Original} - ET_{Proposed}}{ET_{Original}} \times 100(\%) \qquad (20)$$

where $ET_{Original}$ represents the encoding time of the original HTM-16.3 encoder, and $ET_{Proposed}$ is the encoding time of the proposed algorithm.

It can be seen from Table 5 that the proposed algorithm can saves a considerable encoding time for texture and depth maps, and it can provide a similar performance for all test sequences compared to the original 3D-HEVC encoder. The proposed algorithm decreases the encoding time from

TABLE 6. Experimental results of the proposed algorithm compared with related works in 3D-HEVC.

Sequences	[10]		[11]		[12]		[13]		[14]		[15]		Proposed	
	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)
Balloons	0.21	38.24	0.10	22.10	0.80	16.32	0.49	18.50	0.24	20.30	0.69	24.70	0.19	40.11
Kendo	0.31	37.54	0.10	18.80	0.28	16.24	0.18	19.40	0.21	19.90	0.58	22.60	0.21	39.36
Newspaper	0.55	35.01	0.10	20.40	0.66	15.17	0.04	14.10	0.11	18.60	0.87	21.50	0.23	39.93
GT_Fly	0.41	40.28	0.00	24.80	X	X	-0.30	23.30	0.22	20.60	0.31	22.30	0.12	41.23
Poznan_Hall2	0.21	35.92	0.10	29.60	-0.63	17.26	0.46	26.30	0.24	23.80	0.93	30.80	0.14	41.48
Poznan_Street	0.25	35.01	0.10	24.10	0.87	17.04	0.20	15.80	0.21	21.20	0.57	25.40	0.23	40.29
Undo_Dancer	0.25	34.14	-0.1	24.10	X	X	0.01	14.15	0.13	17.70	0.69	20.20	0.28	38.29
Shark	0.42	40.89	0.00	28.60	0.01	16.47	0.28	18.32	0.11	17.40	0.32	19.40	0.32	41.33
1024 × 768	0.39	36.93	0.10	20.40	0.58	15.91	0.23	17.24	0.28	19.60	0.71	22.93	0.21	39.80
1920 × 1088	0.30	37.25	0.00	26.20	0.08	16.92	0.13	19.59	0.18	20.10	0.50	23.62	0.22	40.52
Average	0.34	37.13	0.00	24.10	0.33	16.42	0.17	18.71	0.22	19.90	0.58	23.40	0.22	40.25

TABLE 7. Performance comparison of the proposed algorithm with 3D-HEVC related works.

Works	HTM Software	BD-BR (%)	Time Saving
[10]	16.2	0.34	37.13
[11]	16.2	0.00	24.10
[12]	15.1	0.33	16.42
[13]	16.0	0.17	18.71
[14]	16.0	0.22	19.90
[15]	8.0	0.58	23.40
Proposed	16.3	0.22	40.25

38.29% to 41.48 % and 40.22 % on average for HTM-16.3. Meanwhile, the average decrease of BD-PSNR is 0.01 dB for depth maps and 0.02 dB for texture videos. Moreover, the average BD-BR increase is 0.31% for texture videos and 0.22% for depth maps. The results indicated in Table 5 show that the proposed algorithm based on the AB-NN and GLCM features can avoid unnecessary CU size in 3D-HEVC inter-coding with negligible loss of coding efficiency.

Figure 6 illustrates more detailed experiment results of the proposed algorithm compared to the original 3D-HEVC for four typical sequences: Balloons (1024 × 768), Newspaper (1024 × 768), GT_Fly (1920 × 1088), and Poznan_Hall2 (1920 × 1088) for texture and synthesized views. It can be seen from Figure 6 that the proposed algorithm achieves better RD performance for the four test sequences.

Table 6 compares the proposed algorithm with the views synthesis performance in related works [10]–[15] that also focus on RA configuration. It can be seen from Table 6 that the proposed algorithm can greatly reduce the encoding complexity compared to the inter-coding related works.

When comparing the proposed algorithm with our past work [10], which is focused on a CU size decision on 3D-HEVC inter-coding, it can be seen that our previous work can reduce the encoding time up to 37.13% on average, which is less than the 40.25% in our proposed work, and a 0.34% increase of synthesis BD-BR for HTM-16.2.

Concerning the work proposed in [11], the BD-BR is negligible. Meanwhile, our proposed algorithm performs a better gain in encoding speed. Therefore, it achieves only 24.10% on average relative to HTM-16.2. Thus, our method can save more overall encoding time with the BD-BR that slightly increases in the synthesized views.

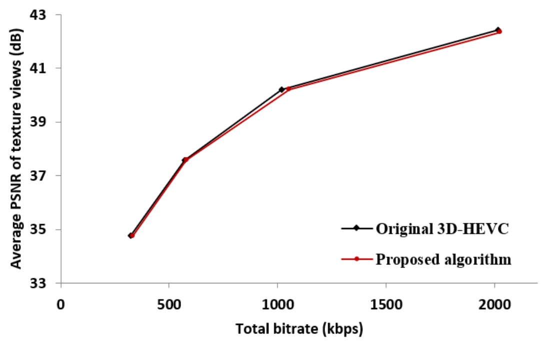
By comparing with the work in [12], our proposed algorithm reduces greatly the complexity of the 3D-HEVC encoder. It can be seen that the authors do not present the results of all test sequences, and their work achieves only a 16.42% time savings on average in correlation with HTM-15.1, with a 0.33% synthesis BD-BR increase.

In [13], the proposed algorithm reduces only 18.71% on average of the encoding time in correlation with HTM-16.0, indicating that our proposed work is better in terms of performance, and BD-BR increase is approximately the same.

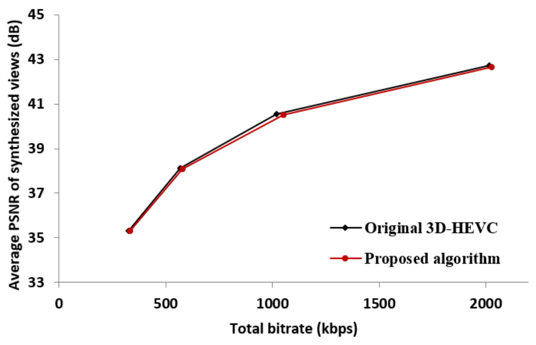
In comparison with the approach in [14], the proposed algorithm achieves the same encoding efficiency with 0.22% BD-BR increase in the synthesized views. Thus, the algorithm presented in [14] achieves a 19.90% reduction in encoding time relative to HTM-16.0 which is less than half of our results.

Compared with the work in [15], the proposed work achieves a 23.40% time savings on average and it has been implemented in the HTM-8.0 version, whereas the method proposed in the present work was implemented on an HTM-16.3 version, which fully implements the final 3D-HEVC standard. Furthermore, in our proposed algorithm, the coding efficiency loss is better, where the average of BD-BR increase in [15] is approximately 0.58%.

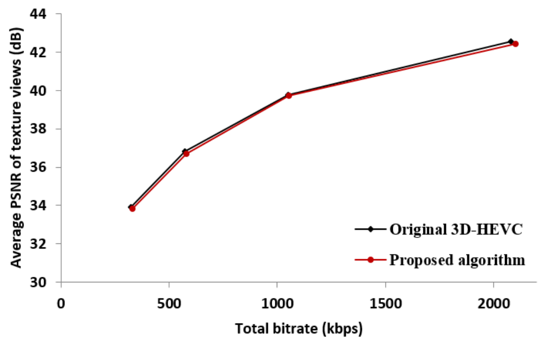
Figure 7 shows the time saving comparison of the proposed algorithm compared to the related works in texture videos,



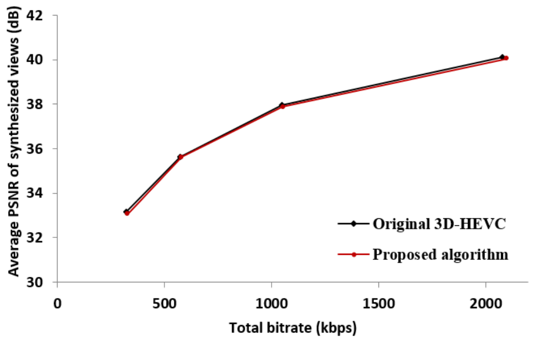
(a) Texture views of Balloons sequence



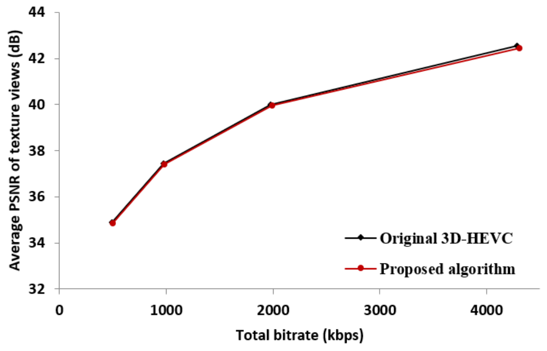
(b) Synthesized views of Balloons sequence



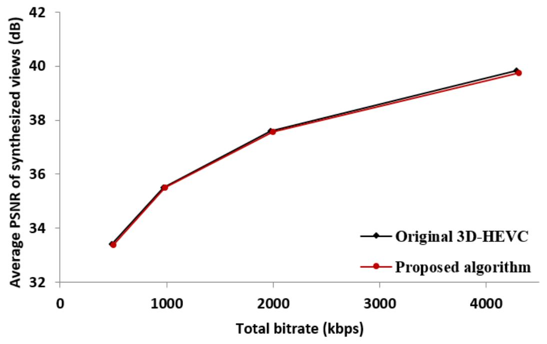
(c) Texture views of Newspaper sequence



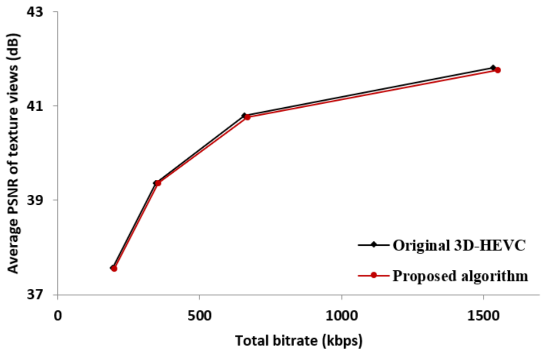
(d) Synthesized views of Newspaper sequence



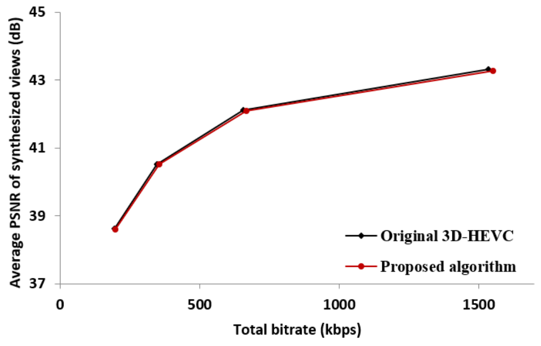
(e) Texture views of GT_Fly sequence



(f) Synthesized views of GT_Fly sequence

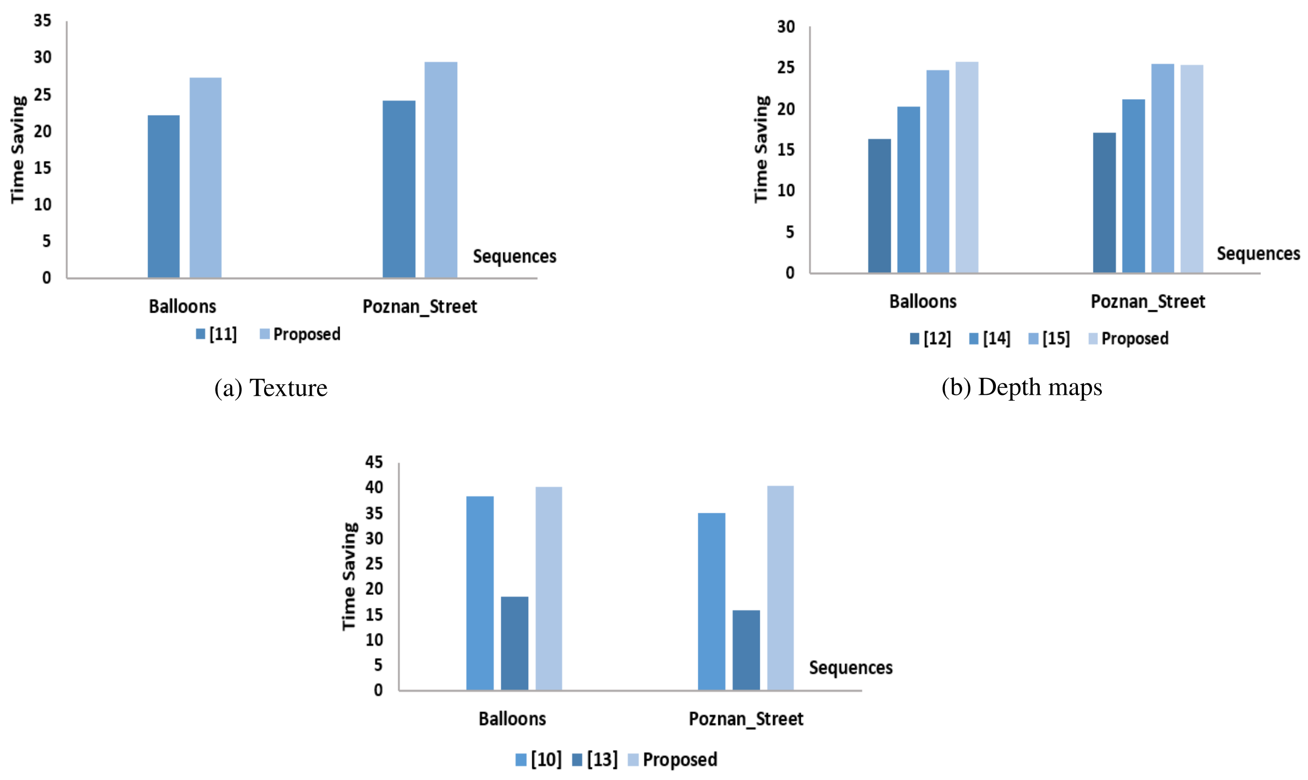


(g) Texture views of Poznan_Hall2 sequence



(h) Synthesized views of Poznan_Hall2 sequence

FIGURE 6. Experimental results of RD curve of the Balloons, Newspaper, GT_Fly and Poznan_Hall2 sequences under different QP combinations for texture video and depth map (25, 34), (30, 39), (35, 42) and (40, 45), RA configuration.



(a) Texture

(b) Depth maps

(c) Texture and depth maps

FIGURE 7. Time saving comparison of the proposed algorithm with related works in texture (a), depth maps (b) and both (c), separately, for Balloons and Poznan_Street sequences.

depth maps, and both, separately for two typical sequences: Balloons (1024 × 768) and Poznan_Street (1920 × 1088). It can be seen from Figure 7 that the proposed algorithm achieves better time saving for the two test sequences compared to the related works in texture views, depth maps views and the both components.

Table 7 presents the comparison of our proposed approach and the related works. The performance of our proposed algorithm in terms of time-saving and coding efficiency is exceptionally high as both texture video and depth maps components are fully employed and an efficient machine learning model is used in this work for early termination of CU encoding in 3D-HEVC.

VII. CONCLUSION

In this paper, we proposed an early termination of the CU encoding algorithm for reducing the complexity of 3D-HEVC texture and depth map inter-coding. The proposed approach is based on first and GLCM features and a boosting neural network for the training model. After extracting the features from the original encoder, we apply the IG model to select the ones with high correlation with the CU partition. The selected features are used in the AB-NN training model to find the suitable thresholds for texture and depth maps. The proposed algorithm can avoid unnecessary CU sizes in the 3D-HEVC encoder. Experimental results show that the proposed approach achieves considerable encoding time savings for 3D-HEVC inter-coding while maintaining negligible coding performance degradation.

REFERENCES

[1] K. Müller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F. Rhee, G. Tech, M. Winken, and T. Wiegand, "3D high-efficiency video coding for multi-view video and depth data," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3366–3378, May 2013.

[2] K. Müller, P. Merkle, and T. Wiegand, "3-D video representation using depth maps," *Proc. IEEE*, vol. 99, no. 4, pp. 643–656, Apr. 2011.

[3] O. Stankiewicz, K. Wegner, and M. Domański, "Study of 3D video compression using nonlinear depth representation," *IEEE Access*, vol. 7, pp. 31110–31122, 2019.

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[5] G. Tech, Y. Chen, K. Müller, J.-R. Ohm, A. Vetro, and Y.-K. Wang, "Overview of the multiview and 3D extensions of high efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 35–49, Jan. 2016.

[6] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block partitioning structure in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1697–1706, Dec. 2012.

[7] S. Bakkouri, A. Elyousfi, and H. Hamout, "Fast CU size and mode decision algorithm for 3D-HEVC intercoding," *Multimedia Tools Appl.*, vol. 79, nos. 11–12, pp. 6987–7004, 2019.

[8] S. Ahn, B. Lee, and M. Kim, "A novel fast CU encoding scheme based on spatiotemporal encoding parameters for HEVC inter coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 3, pp. 422–435, Mar. 2015.

[9] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An effective CU size decision method for HEVC encoders," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 465–470, Feb. 2013.

[10] S. Bakkouri and A. Elyousfi, "Machine learning-based fast CU size decision algorithm for 3D-HEVC inter-coding," *J. Real-Time Image Process.*, vol. 18, no. 3, pp. 983–995, Jun. 2021.

[11] Y. Li, G. Yang, Y. Zhu, X. Ding, Y. Song, and D. Zhang, "Hybrid stopping model-based fast PU and CU decision for 3D-HEVC texture coding," *J. Real-Time Image Process.*, vol. 17, no. 5, pp. 1227–1238, Oct. 2020.

[12] Y.-W. Liao, M.-J. Chen, C.-H. Yeh, J.-R. Lin, and C.-W. Chen, "Efficient inter-prediction depth coding algorithm based on depth map segmentation for 3D-HEVC," *Multimedia Tools Appl.*, vol. 78, no. 8, pp. 10181–10205, Apr. 2019.

[13] J. Chen, B. Wang, J. Liao, and C. Cai, "Fast 3D-HEVC inter mode decision algorithm based on the texture correlation of viewpoints," *Multimedia Tools Appl.*, vol. 78, no. 20, pp. 29291–29305, Oct. 2019.

[14] Q. Zhang, N. Zhang, T. Wei, K. Huang, X. Qian, and Y. Gan, "Fast depth map mode decision based on depth–texture correlation and edge classification for 3D-HEVC," *J. Vis. Commun. Image Represent.*, vol. 45, pp. 170–180, May 2017.

[15] M. Chen, Y. Yang, Q. Zhang, X. Zhao, X. Huangb, and Y. Gan, "Low complexity depth mode decision for HEVC-based 3D video coding," *Optik*, vol. 127, no. 11, pp. 4758–4767, Jun. 2016.

[16] (2018). *Joint Collaborative Team on 3D Video Coding (JCT-3V) HTM 16.3 Reference Software*. [Online]. Available: https://hevc.hhi.fraunhofer.de/trac/3d-hevc/browser/3DVCSoftware/tags/HTM-16.3

[17] K. Mueller and A. Vetro, *Common Test Conditions of 3DV Core Experiments*, document JCT3V-G1100, 7th Meeting, Joint Collaborative Team on 3D Video Coding Extensions (JCT-3V), San Jose, CA, USA, 2014.

[18] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, Jul. 1990.

[19] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1995.

[20] R. Schapire, "Theoretical views of boosting," in *Proc. Eur. Conf. Comput. Learn. Theory*, 1999, pp. 1–10.

[21] G. Rätsch, T. Onoda, and K. Müller, "Soft margins for AdaBoost," *Mach. Learn.*, vol. 42, no. 3, pp. 287–320, 2001.

[22] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, no. 2, pp. 139–157, Aug. 2000.

[23] H. Schwenk and Y. Bengio, "Boosting neural networks," *Neural Comput.*, vol. 12, no. 8, pp. 1869–1887, 2000.

[24] H. Schwenk and Y. Bengio, "AdaBoosting neural networks: Application to on-line character recognition," in *Proc. Int. Conf. Artif. Neural Netw.*, 1997, pp. 967–972.

[25] M. M. Baig, M. M. Awais, and E. S. M. El-Alfy, "AdaBoost-based artificial neural network learning," *Neurocomputing*, vol. 248, pp. 120–126, Jul. 2017.

[26] Y. Kim, "Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size," *Expert Syst. Appl.*, vol. 34, no. 2, pp. 1227–1234, Feb. 2008.

[27] M. W. Ahmad, M. Mourshed, and Y. Rezgui, "Trees vs neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption," *Energy Buildings*, vol. 147, pp. 77–89, Jul. 2017.

[28] S. Haykin and S. Hakin, *The Neural Networks a comprehensive Foundation*. New York, NY, USA: Macmillan College Publishing Company, 1999.

[29] S. Bakkouri and A. Elyousfi, "Effective CU size decision algorithm based on depth map homogeneity for 3D-HEVC inter-coding," in *Proc. Int. Conf. Intell. Syst. Comput. Vis. (ISCV)*, Jun. 2020, pp. 1–6.

[30] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Fast 3D-HEVC depth map encoding using machine learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 3, pp. 850–861, Mar. 2020.

[31] R. W. Grubbström and O. Tang, "The moments and central moments of a compound distribution," *Eur. J. Oper. Res.*, vol. 170, no. 1, pp. 106–119, Apr. 2006.

[32] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.

[33] D. J. Marceau, P. J. Howarth, J. M. Dubois, and D. J. Gratton, "Evaluation of the grey-level co-occurrence matrix method for land-cover classification using spot imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 28, no. 4, pp. 513–519, Jul. 1990.

[34] S. Jadhav, H. He, and K. Jenkins, "Information gain directed genetic algorithm wrapper feature selection for credit rating," *Appl. Soft Comput.*, vol. 69, pp. 541–553, Aug. 2018.

[35] L. Li, H. Liu, Z. Ma, Y. Mo, Z. Duan, J. Zhou, and J. Zhao, "Multi-label feature selection via information gain," *Adv. Data Mining Appl.*, pp. 345–355, 2014.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[37] G. Bjontegaard, "Calculation of average PSNR difference between RD-curves," in *Proc. 13th VCEG-M33 Meeting*, Austin, TX, USA, Apr. 2001.

[38] G. Bjontegaard, *Improvements of the BD-PSNR Model*, document VCE-GAI11, 35th VCEG Meeting, Berlin, Germany, 2008.

[39] M. Tanimoto, T. Fujii, and K. Suzuki, *View Synthesis Algorithm in View Synthesis Reference Software 2.0 (VSRS2.0)*, document ISO/IEC JTC1/SC29/WG11 M16090, Lausanne, Switzerland, 2008.

SIHAM BAKKOURI received the Engineering degree in computer science from the National Engineering School of Applied Sciences, Ibn-Zohr University, Agadir, Morocco, in 2017. She is currently pursuing the Ph.D. degree with the Faculty of Sciences, Ibn-Zohr University. Her current research interests include complexity reduction algorithms, image and video compression, machine learning for video coding, and future video coding standards.

ABDERRAHMANE ELYOUSFI received the M.S. degree in computer, telecommunication and multimedia from the Faculty of Sciences, University Mohammed V-Agdal, Rabat, Morocco, in 2004, the Ph.D. degree in computer and telecommunication from University Mohammed V-Agdal, in 2009, and the H.D.R. Ph.D. degree. From 2010 to 2015, he worked as an Assistant Professor at the National Engineering School of Applied Sciences, University Ibn-Zohr, Agadir, teaching courses related to the computer sciences field. Since 2015, he has been working as a Full Professor at the National Engineering School of Applied Sciences, Ibn-Zohr University, teaching computer sciences. His main research interests include video coding, pattern recognition, image processing, and telecommunications.