

Received July 6, 2020, accepted July 29, 2020, date of publication August 4, 2020, date of current version August 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014163

# Decision Tree Models and Early Splitting Termination in Screen Content Extension of High Efficiency Video Coding

EMAD BADRY<sup>1,4,5</sup>, (Graduate Student Member, IEEE), KOJI INOUE<sup>1,2</sup>, (Member, IEEE), AND MOHAMMED SHARAF SAYED<sup>3,4</sup>, (Member, IEEE)

<sup>1</sup>E-JUST Center, Kyushu University, Fukuoka 819-0382, Japan

<sup>2</sup>Department of Advanced Information Technology, Kyushu University, Fukuoka 819-0382, Japan

<sup>3</sup>Department of Electronics and Communications Engineering, Zagazig University, Zagazig 44519, Egypt

<sup>4</sup>Department of Electronics and Communications Engineering, Egypt-Japan University of Science and Technology, Alexandria 21934, Egypt

<sup>5</sup>Department of Electrical Engineering, Suez Canal University, Ismailia 41522, Egypt

Corresponding author: Emad Badry (emad.mahmoud@ejust.edu.eg)

This work was supported in part by the Egypt Japan University of Science and Technology (E-JUST) and in part by Kyushu University.


**ABSTRACT** A Screen Content (SC) extension to the High Efficiency Video Coding (HEVC) standard has been developed to improve the encoding of SC sequences. SC scenes are rich in repeated patterns, non-noisy regions, sharp-edge areas, and blocks with limited colors, which differ from Natural Content (NC) videos. For the SC Coding (SCC) process, new tools have been incorporated into the HEVC SC extension such as Intra Block Copy (IBC) and Palette (PLT); these tools improve the compression accuracy at the expense of high computational complexity. In this paper, we present a framework to reduce the encoding time of SC encoders by exploiting the characteristics of the SC blocks. The framework contains two techniques. The first is called Decision Tree Models (DTM), and it includes decision tree-based classification blocks to reduce the number of executed modes. In the DTM technique, the features of each Coding Unit (CU) are extracted and trained to build the classification trees. To further speed up the encoding process, a second technique called Early Splitting Termination (EST) is suggested to stop the normal splitting process of homogeneous blocks by measuring the luminance contrast inside the blocks. Compared with the HM-16.7+SCM-6 reference test model, the proposed framework can provide a 35.96% encoding time reduction on average with only a 0.89% increase in Bjontegaard Delta bit-rate (BD-Rate) under the All-Intra (AI) configuration profile, which outperforms the approaches in the literature. In addition, the proposed framework reduces the encoding time by 54.3% on average for a number of NC sequences recommended for conventional HEVC test, with only 0.74% increment in the BD-Rate. For further speeding up, the proposed scheme has been integrated with an existing approach. Consequently, a 45.84% reduction in time complexity is obtained with a BD-Rate increase of only 1.3%.

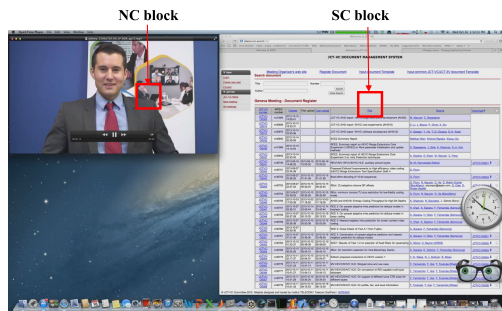
**INDEX TERMS** Screen content coding (SCC), intra block copy (IBC), palette (PLT) mode, decision tree.

## I. INTRODUCTION

With the spread of communication networks and computer technologies, many video applications have grown such as online-education, video conferences, documents, and slides sharing. These applications handle different types of videos such as camera-captured, graphics, and mixed videos. Mixed videos have natural and synthetic scenes. The High Efficiency

Video Coding (HEVC) [1] is a video coding standard used to efficiently compress the videos. It was jointly developed by the ITU-T Video Coding Expert Group (VCEG) and ISO/IEC Motion Picture Expert Group (MPEG). HEVC can achieve a 50% reduction in bit-rate with respect to the previous standard H.264/AVC [2] at the same video quality. The conventional HEVC standard is suitable to efficiently compress the sequences captured by cameras, and these types of video are called Natural Content (NC) sequences. On the other hand, Screen Content (SC) sequences, which are generated

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



**FIGURE 1.** NC and SC block examples from the first frame in the “MissControlClip2” sequence.

by computers, demand additional tools to handle their special characteristics. In SC videos, the blocks have a small number of distinct colors, frequently repeated patterns, non-noisy regions, and many areas with complex edges. Fig. 1 illustrates an example of NC and SC blocks. In 2014, a proposal was called by the Joint Collaborative Team on Video Coding (JCT-VC) for HEVC SC extension. As a result, the HEVC SC extension [3]–[5] was introduced to encode SC videos. It was finalized in 2016.

In intra coding, new modes have been introduced into the SC extension such as the Intra Block Copy (IBC) [6]–[10] and Palette (PLT) modes [11]–[13] to give significant improvement in the coding performance. The IBC mode is similar to the motion estimation-compensation mode in traditional HEVC [14]–[16]. In motion estimation-compensation, the current block tracks the best matched blocks across the temporal frames. However, IBC conducts this search in the current frame only. In the PLT mode, the Coding Unit (CU) encoding is performed by assigning each pixel inside the block with an index. The palette table that includes the indices is transmitted to the decoder for the reconstruction process. In spite of improved coding performance brought by using the SCC tools, the computational complexity increases because of the heavy computations performed by these tools.

To overcome the computational complexity problems of the SCC tools, many studies have been conducted. In [17], the check of the IBC mode is skipped completely if the cost of the conventional intra mode is smaller than a threshold and the CU size is larger than  $16 \times 16$ . IBC is conducted along 1-D search regions for  $8 \times 8$  pixels blocks if the gradient value is smaller than a pre-defined threshold. The authors in [8] speed up the SC encoders by restricting the searching process of the IBC mode to the blocks that have the same hash key. New hash key formulas were suggested for Prediction Unit (PU) sizes of  $8 \times 4$ ,  $4 \times 8$ , and  $16 \times 6$  pixels, the PU is the smallest partitioning unit in HEVC. The CUs are categorized as smooth blocks or sharp blocks in [18], which stops CU segmentation early to increase SC compression speed. In [19], the CUs are classified into background and foreground. A CU is considered as a stationary CU if the Sum of Absolute Difference (SAD) between the current CU and its corresponding CU in previous frames is smaller than

a threshold; intra and IBC modes are skipped for background CUs. In addition, the authors suggested a scheme to reduce the searching time in the IBC mode by changing the step size adaptively. In [20], the CUs are categorized into NC or SC in accordance with the statistics of the contents; IBC and PLT modes are ignored for NC blocks if the best mode of the intra mode is DC or PLANAR. For SC blocks, all the modes are executed. Also, the depth information of the neighbor CUs and the coding bits are used to make a fast size decision. The pixel exactness value was used in [21] to determine if the SCC tools could be skipped. Furthermore, it was used to determine if CU pruning could be terminated early, and the order of the PLT and IBC modes was swapped. The IBC would then be activated in accordance with the Rate-Distortion (RD) cost of the PLT mode.

The previous works in the literature can be categorized as conventional methods. A number of previous work have used Machine Learning (ML) techniques in their schemes. Huang *et al.* [22] designed a traditional neural network to classify the CUs into NCs or SCs to bypass modes in each category, where it gets depth information from adjacent CUs to utilize a fast CU size decision. Bayesian rule classification technique was used in [23] and [24]. In [23], the regions are classified into textual or pictorial ones by using the corner point detection method. Consequently, an early mode skipping method is utilized. Kuang *et al.* [24] built a Bayesian rule model for a fast mode and CU size decision. It exploits the optimal mode information of spatial neighbor CUs to utilize further mode skipping. In [25], the characteristics of the CU, the information from neighboring CUs, and the intermediate cost information are learned by using the random forest method to build a fast mode decision framework. A Decision Tree (DT) classification approach was used in [26] and [27] to reduce the time complexity of SC encoders. In [26], the author designed two classification trees, one to determine whether the coding modes are checked for the current depth, one to classify the blocks into NC or SC blocks. Intra mode is skipped when the CU is considered as SC, while SC tools are skipped for NC CUs. The authors in [27] proposed a fast intra coding framework. In this, a DT is inserted before each mode to facilitate each mode separately. The intermediate information between the modes was considered during the training step. Other works utilize DTs to reduce the computational complexity of traditional HEVC, such as [28] and [29]. The goal of the method in [28] is to terminate the pruning process of the CUs early to eliminate unnecessary mode checking. While the main contribution of [29] is to make the conventional intra mode work faster, neither [28] nor [29] utilize the SCC modes. So, they are unsuitable for SC encoders.

In this paper, we propose a framework to reduce SC encoding time by achieving a fast CU mode decision and by ending the partitioning of smooth CUs early. The mode skipping is performed by classifying the CUs into NCs and SCs. Intra mode is conducted for NC CUs and SC CUs ignore the testing of intra mode to reduce the encoding time. Moreover, before

the classifiers, intra, IBC, and PLT modes can be skipped completely in accordance with a predictive model inserted before intra mode. To achieve more time-saving for SC CUs, DTs are inserted before the PLT mode to determine if the check of IBC mode is sufficient or the PLT mode should also be checked. To get a fast CU size decision, the splitting of homogeneous CUs is terminated early to reduce the computational complexity. CU smoothness can be checked by estimating the difference between the maximum and minimum luminance inside the block. A threshold is suggested to determine a limit since when the luminance contrast is smaller than it, the splitting process is skipped.

A DT classification approach is used here as in [26]–[29] due to its simplicity at the training and implementation phases, which makes the real implementation applicable. This is different from the work of [22] that adopts a traditional neural network with high computational complexity. Compared with [25] and [27], our framework guarantees that at least one mode should be skipped, where the classifiers ensure that not all the modes can be checked for a CU. However, the existing schemes in [25] and [27] utilize each mode separately. Thus, in some cases, all the modes may be executed. We suggest an Early Splitting Termination (EST) technique to further speed up the encoder, which was not adopted in [19], [25], and [27]. Our framework does not need to access the adjacent CUs as that in [20] and [25], which reduces the memory requirements. The correlation between the optimal mode of the current CU and its parent before the splitting is newly suggested here as a feature to improve CU classification accuracy. The experimental results show that the proposed framework could reduce the encoding time with negligible quality degradation.

The rest of the paper is organized as follows. Section II explains the mode decision process of the SC encoder. Also, it studies a statistical analysis. Section III gives details about the proposed framework. The experimental results are discussed in Section IV. Finally, the paper is concluded in Section V.

## II. SCREEN CONTENT INTRA CODING

In this section, we explain the mode decision process of SC encoders and that for intra coding to get the optimal mode and size. Also, statistical analysis regarding the mode decision process will be presented.

### A. MODE DECISION PROCESS

The HEVC SC extension handles videos in the same way as traditional HEVC. The input frame is divided into non-overlapped square blocks. The basic unit is called a Coding Tree Unit (CTU), which is  $64 \times 64$  pixels (depth 0) by default. The SC encoder further divides the CTU recursively into four smaller equal CUs. For each particular  $2N \times 2N$  CU,  $N$  can be 32, 16, 8, or 4. The splitting process continues until the CUs are  $8 \times 8$  (depth 3). An example of CTU partitions and their corresponding quad-tree structure is shown in Fig. 2. A CU is further divided into one, two, or four PUs. To find the

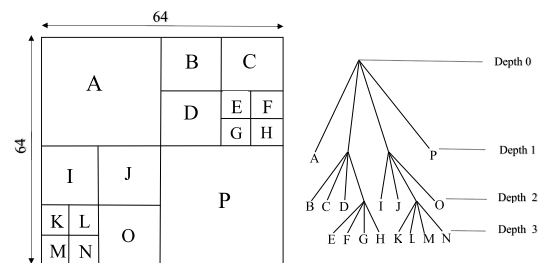


FIGURE 2. Example of CTU partitions (left) and their corresponding quad-tree structure (right).

best structure of the CTU, the encoder compares the RD cost of each CU with the sum of the RD costs of its four sub-CUs.

In All-Intra (AI) mode, the RD cost is obtained for each CU by conducting intra, IBC, and PLT modes. Then, the mode with the least RD cost among them is chosen as the optimum mode. The evaluated RD cost  $J_{mode}$  is estimated as follows:

$$J_{mode} = D_{mode} + \lambda \times B_{mode} \quad (1)$$

where  $D_{mode}$  denotes the distortion between current CU and its reconstructed CU,  $\lambda$  is the Lagrangian multiplier that depends on the quantization parameter (QP), and  $B_{mode}$  is the actual number of bits used to signal the CU coding information.

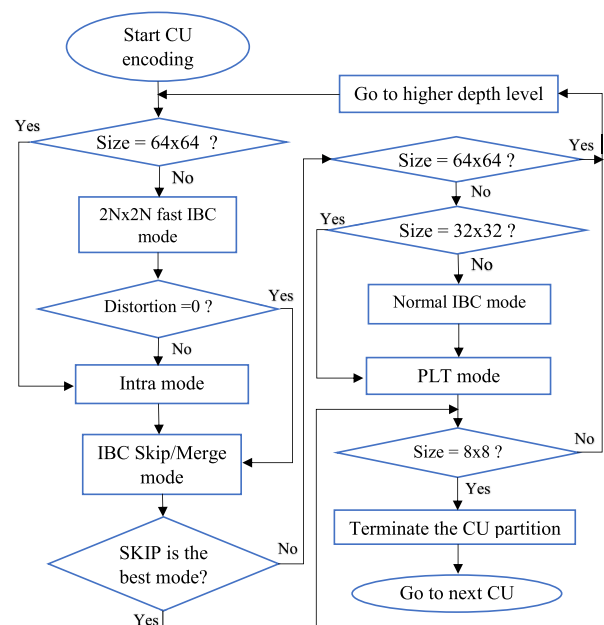


FIGURE 3. Mode decision process of the SC encoder.

The details of the mode decision process are illustrated in Fig. 3. First, the CUs that have sizes less than  $64 \times 64$  pixels locate the best matched block by conducting the  $2N \times 2N$  fast IBC mode through a candidate list of Block Vectors (BVs). These BVs are counted up to 64 and are constructed from the last two coded CUs and the neighboring CUs. Due to the small number of candidates, this process does not clearly

affect the time complexity. The term “2N×2N fast IBC” is used here to differentiate it from “normal IBC”, which will be described later. If the best matched block is found with zero distortion, intra mode is skipped. Otherwise, the conventional intra mode is conducted for all CU sizes, where it calculates RD costs for 35 modes [30], including 33 directional modes, DC mode, and planar mode to determine the best RD cost among them. After checking intra mode, IBC Skip/Merge mode is checked by using BV predictors. If the Skip mode is the best mode at this point or if the CU size is  $64 \times 64$ , the SCC tools are ignored, otherwise, the SC tools are tested. The repeated patterns usually appear in smaller CUs more than in larger ones. Therefore, normal IBC is applied to the CU sizes of  $16 \times 16$  and  $8 \times 8$ . The normal IBC has two search strategies, where the current PU can locate the best matched block either by running a search within a pre-defined search area or by using a hash key search method. For the first method, the  $16 \times 16$  blocks carry out the searching process within 1-D vertical and horizontal regions along the current frame. For the  $8 \times 8$  CUs, the search range is limited to the current CTU and left CTU. Furthermore,  $2N \times 2N$  and  $2N \times N$  PUs can search along 1-D or 2-D areas while the search range of the  $N \times 2N$  PU is restricted to the 1-D areas. When it comes to the hash key based search, this method is implemented for the  $8 \times 8$  CUs with  $2N \times 2N$  PUs. For each CU, a unique hash key is estimated, and to calculate the key, the CU is partitioned into four equal parts. Following that, the DC value of each part and the gradient value of the CU are estimated to form a 16-bit key. The block matching of the hash key search is conducted for the CUs that have the same hash key inside the current frame. More details about the IBC search strategies can be found in [8] and [21]. When the IBC test is finished, the PLT mode is checked if the CU size is less than  $64 \times 64$  pixels. After checking all modes, the best mode is the one that has minimum RD cost. If the CU is larger than  $8 \times 8$ , it is divided into four smaller CUs. Then, the mode decision process is repeated again for each CU.

The additional complexity brought by introducing the SCC tools over the conventional intra mode is analyzed by conducting a simulation using the HEVC SC test model version (HM-16.7+SCM-6) [31], hereafter SCM-6 for simplicity. The QPs of 22, 27, 32, and 37 were used under the All-Intra (AI) configuration profile. The test sequences that are recommended by Common Test Conditions (CTC) [32] are used. The test sequences are categorized into four groups: text and graphic with motion (TGM), animation (A), camera-captured content (CC), and (Mixed). Table 1 shows the Bjontegaard Delta bit-rate (BD-Rate) [33] and the Time Saving (TS) of the SCM-6 version without the SCC tools compared with the same version with SCC enabled, TS is estimated as follows:

$$TS = \left\{ \frac{Time_{conv} - Time_{new}}{Time_{conv}} \right\} \quad (2)$$

where  $Time_{new}$  is the encoding time with modifications that represents the encoder with SCC disabled,  $Time_{conv}$  is the

encoding time of the conventional encoder with the SCC tools enabled.

**TABLE 1. BD-Rate(%) and TS(%) of SCM-6 with SCC disabled compared with SCM-6 with SCC tools activated.**

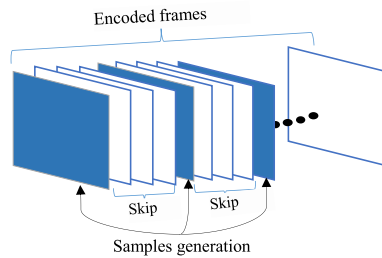
Sequences	Type	BD-Rate (%)	TS (%)
WebBrowsing	TGM	225.95	50.87
Map	TGM	25.99	66.15
Programming	TGM	93.55	55.5
Desktop	TGM	253.19	53.92
SlideShow	TGM	32.23	48.76
FlyingGraphics	TGM	137.17	61.96
ChineseEditing	TGM	72.61	68.55
Console	TGM	184.91	45.69
BasketballScreen	Mixed	70.27	62.8
MissionControlClip2	Mixed	40.24	65.75
MissionControlClip3	Mixed	82.16	60.41
Robot	A	1.47	65.21
Kimono1	CC	-0.08	51.98
<b>Average</b>		<b>93.82</b>	<b>58.27</b>

Table 1 shows that by skipping IBC and PLT modes, the encoding time is reduced by 58.27% on average and up to 68.55%. Also, the BD-Rate increases to 93.82% on average, which indicates that the coding performance is highly degraded. Disabling the SCC tools affects the TGM and Mixed videos clearly compared with the A and CC types. For the A and CC types, the encoding time is reduced by 65.21% and 51.98%, respectively, with a small performance degradation for category A, where the BD-Rate is 1.47% only, and the BD-Rate is -0.08% for CC sequence. We can state that the videos that have many SC blocks can be efficiently encoded by IBC and PLT, and the conventional intra mode is sufficient to encode the A and CC types well.

## B. STATISTICS ANALYSIS

To get data samples for analysis and training, the original SCM-6 software was modified to extract a group of features for each CU. These features will be discussed later. It was run under the AI configuration profile for the QPs of 22, 27, 32, and 37. The first 100 frames of eight selected sequences from Table 1 were encoded, where the eight videos cover the different characteristics. The test sequences are “WebBrowsing”, “Slideshow”, “Desktop”, and “Console” as the TGM type, “BasketballScreen” and “MissionControlClip2” as the Mixed type, and “Robot” and “Kimono1” as the A and CC types, respectively. To reduce the redundant data, the first frame of each group of four frames was used to extract the samples, while the other three frames were skipped as shown in Fig. 4. Collected data samples are used for all analysis and training in this work.

Table 2 tabulates the mode distributions at a QP of 22. From Table 2, we can notice that the distributions vary in accordance with category and CU size. TGM videos are rich in SC regions, so, it is observed that the majority of the CUs are encoded by using the SCC tools and reach up to 68.55% for  $8 \times 8$  CUs. However, for  $64 \times 64$  CUs, intra mode encodes most of the CUs because the intra and IBC



**FIGURE 4.** The method of extracting data samples, blue frames are used to generate the samples.

**TABLE 2.** Mode distributions of each CU size for the data samples.

Type	CU size	Intra (%)	IBC (%)	PLT (%)
TGM	64x64	84.46	15.54	0
	32x32	18.22	19.59	62.19
	16x16	22.04	49.08	28.88
	8x8	31.45	57.93	10.62
Mixed	64x64	83.08	16.92	0
	32x32	43.19	21.06	35.75
	16x16	47.1	40.69	12.21
	8x8	56.27	40	3.73
A+CC	64x64	100	0	0
	32x32	99.32	0.14	0.54
	16x16	98.59	1.11	0.3
	8x8	98.01	1.91	0.08

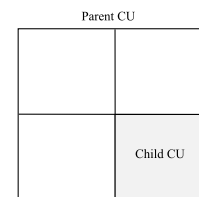
Skip/Merge modes are the only modes that are tested at this size. The Mixed type includes natural and synthetic blocks, so the number of CUs that take intra mode as the best mode are higher than those of the TGM type. For the A and CC types, intra mode is selected to encode CUs in almost 100% of all cases. From the presented data, we can conclude that the SCC tools can efficiently encode SC blocks with low cost, and natural or animated videos can be well-compressed by using the conventional intra mode. Consequently, using the SCC tools to encode these types will not improve performance and waste encoding time. Therefore, by predicting the CU type before the RD Optimization (RDO) process, the computational complexity can be reduced.

As mentioned before, there are an abundance of repeated patterns in SC scenes, which can be exploited by the 2Nx2N fast IBC mode to reduce the number of RDO computations. This is because the 2Nx2N fast IBC mode finds the best matching block at a low cost because the number of candidates are small, and the BV predictors come from the neighbors and the last coded BVs only. Therefore, for most SC CUs, the best IBC cost resulted from 2Nx2N fast IBC. Table 3 tabulates the percentages of CUs that are encoded by IBC mode, and the minimum cost was detected after the encoder had searched the 2Nx2N fast IBC. The data are taken at a QP of 22 for the CU sizes of 8 × 8, 16 × 16, and 32 × 32. From Table 3, we can notice that the majority of CUs that are encoded by IBC can find the best cost before the normal IBC search, especially for the larger blocks. The percentages are smaller for 8 × 8 CUs because there are many search strategies that are implemented for this size to

**TABLE 3.** Ratio of CUs with the best RD cost resulted from 2Nx2N fast IBC from all CUs encoded by IBC mode.

Sequences	Type	8x8	16x16	32x32
Webbrowsing	TGM	62.15%	95.89%	96.80%
Slideshow		58.33%	97.68%	97.21%
Console		55.71%	98.71%	98.29
Desktop		56.88%	95.76%	93.44%
BasketballScreen	Mixed	53.52%	96.13%	94.11%
MissionControlClip2		47.53%	96.18%	93.16%
Robot	A	99.30%	97.76%	100%
Kimono1	CC	96.89%	96.37%	100%

search the matched blocks as discussed before. For 32 × 32 CUs, the best IBC cost come from the 2Nx2N fast IBC and IBC Skip/Merge modes only, where the percentages are close to 100%. In conclusion, for many SC CUs, the 2Nx2N fast IBC mode may be sufficient to encode CUs with a smaller complexity and that can be satisfied by observing the RD cost of the 2Nx2N fast IBC mode since a small RD cost gives an insight in that the subsequent modes can be skipped without clearly affecting the performance.



**FIGURE 5.** A Child CU and its Parent CU at a certain depth.

The type of a CU at  $depth_i$  highly correlates to the type of larger CU at  $depth_{i-1}$ , where  $i$  is the depth value,  $i \in \{1,2,3\}$ . “Parent CU” denotes the CU at  $depth_{i-1}$  and “Child CU” denotes the CU at depth  $i$  in this work. Fig. 5 shows a Child CU and its Parent CU at a certain depth. The CU type is considered as NC if it is encoded by the conventional intra mode. On the other hand, when the IBC or PLT mode is chosen as the optimum mode, the CU is SC. To validate the correlation between the CUs and their Parent CUs, Fig. 6 illustrates the percentages of CUs that have the same type as their Parent CU. The types of each CU and their Parent CU were collected previously for the analysis. Fig. 6a shows the percentages for when the Child CUs are NC. The figure shows that 88.8% of 8 × 8 CUs and their Parent CU are NC type. The high correlation can be also shown in the 16 × 16 case with 87.9%. For the 32 × 32 blocks, this result cannot be confirmed because the Parent CUs, which are 64 × 64 pixels, are checked only by intra or IBC Skip/Merge modes. Similarly, Fig. 6b shows the statistics of SC blocks, in which, the percentages of the 8 × 8 and 16 × 16 blocks are 86.6% and 86.2%, respectively. From this data, we can conclude that there is a high correlation between the types of the Child CUs and their Parent CU, and that can be well-analyzed for smaller CUs. As a consequence, knowing the type of Parent CU will help to limit the number of modes that should be checked for the current CU.

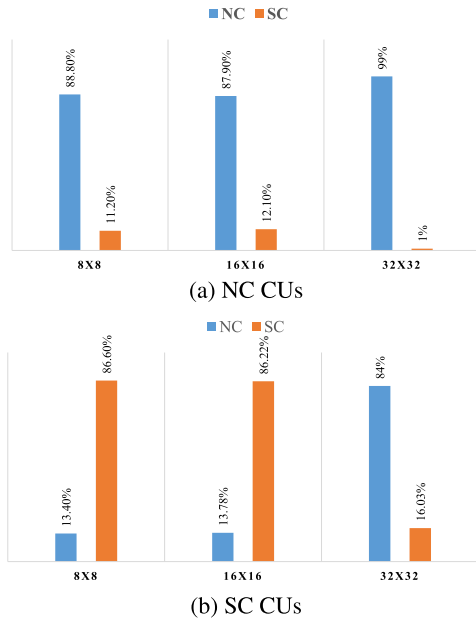


FIGURE 6. Percentages of type similarity between CUs and their Parent CUs. (a) NC CUs (b) SC CUs.

CUs with high degree of textual complexity tend to be split into smaller CUs since they can find similar CUs broadly rather than a larger CUs. In other words, when the block finds a perfect matched block, the difference between the current CU and its predicted one will give a small distortion. As a result, the block is encoded by a fewer number of bits. However, the improvement in compression efficiency comes at the expense of huge computations, which increases the encoding time. On the other hand, smooth CUs have the chance to be efficiently coded without splitting. The smoothness of the CU can be represented by estimating the difference between the maximum and minimum luminance value inside the CU, which is called the Luminance Range ( $LR$ ), defined as in [34]:

$$LR = L_{max} - L_{min} \tag{3}$$

where  $L_{max}$  and  $L_{min}$  represent the maximum and minimum luminance values, respectively. For homogeneous CUs, the luminance values are close to each other or the  $LR$  value is small. To investigate the behavior of the CUs versus the  $LR$  parameter regarding the splitting process, the data samples were partitioned into two classes, and that for the CUs that have sizes more than  $8 \times 8$ . The classes are called Split (SP) and Non-Split (NSP), and CUs were recorded as NSP when the optimum RD cost is smaller than the sum of RD costs of its sub-CUs, and the opposite is true for the SP class. Fig. 7 illustrates the percentages of SP and NSP CUs versus the  $LR$  parameter for  $16 \times 16$  blocks as an example. The test range is between 0 and 127, and the percentages of each range of 16 are grouped together. The number of test samples is 2,390,991 blocks, and they are divided equally between the SP and NSP classes. From Fig. 7, we can notice that the NSP blocks predominate the SP CUs as the  $LR$  decreases,

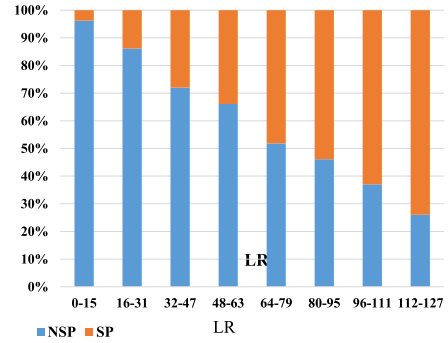
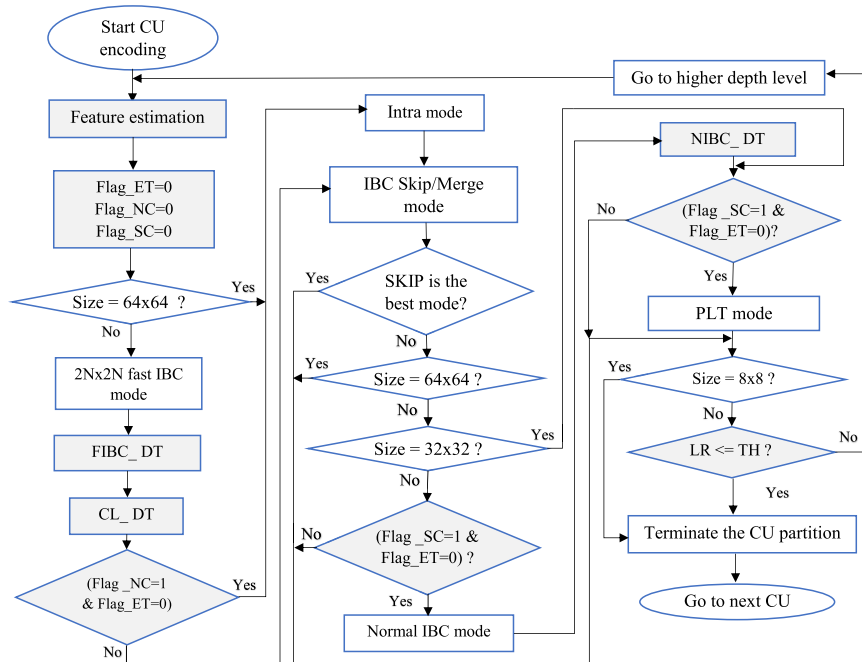


FIGURE 7. Percentages of SP and NSP blocks for  $16 \times 16$ -pixels CUs versus  $LR$ .

reaching up to about 97% for  $LR$ s smaller than 16. As the  $LR$  increases to a higher value, the majority of CUs are recorded as SP. Thus, we can terminate the splitting process of smooth CUs early to reduce the encoding time. The reduction in time complexity comes due to the reduction of the exhaustive search process, since after splitting, the encoder has to check all the modes for all sub-CUs.

### III. PROPOSED FRAMEWORK

As mentioned before, incorporating the SCC tools with the conventional intra mode in the SC extension clearly impacts the time complexity. Therefore, we suggest a framework to reduce the processing time of the SCC. The framework has two techniques in accordance with the previous analysis. The first called Decision Tree Models (DTM) is mainly used to skip the check of unwanted modes, and depends on the DT classification method. The technique consists of three predictive models with each model built by using classification trees. The first model is called  $FIBC\_DT$  and is located after the  $2N \times 2N$  fast IBC mode. When the execution of  $2N \times 2N$  fast IBC mode is finished, the model determines whether to skip the intra, normal IBC, and PLT modes or not. The  $FIBC\_DT$  model works for the CUs of  $8 \times 8$ ,  $16 \times 16$ , and  $32 \times 32$  since the  $2N \times 2N$  fast IBC mode is not employed for the CUs of  $64 \times 64$  pixels. To reduce the encoding time further, the CUs are categorized into NC or SC. Intra mode is conducted for NC CUs where the IBC and PLT modes are activated only when the CU is classified as SC. The model is called  $CL\_DT$  and classifies all the CUs smaller than  $64 \times 64$ . This model resides before the run of the intra, normal IBC, and PLT modes. To further speed up the encoding of the SC CUs, a third model called  $NIBC\_DT$  is placed after the normal IBC block and used to control the activation of the PLT mode by determining if the test of normal IBC is sufficient or not. This predictive model is used for the  $8 \times 8$  and  $16 \times 16$  blocks because normal IBC is performed for these sizes as described before. The second technique in our framework is called Early Splitting Termination (EST), which terminates the division process of homogeneous CUs. Since, the best CU size is determined by checking the  $LR$  value defined in Equ. 3. Then, the splitting process is skipped if the  $LR$  value is smaller than



**FIGURE 8.** Mode decision process by using proposed techniques. The modifications are highlighted in gray.

a threshold TH. Fig. 8 describes the mode decision process by using the proposed techniques, and the modifications are highlighted in gray. This process is different from the mode decision of the conventional encoder shown in Fig. 3. The features of each CU are estimated at the beginning of the CU encoding. There are three flags set by the predictive models: *Flag\_ET*, *Flag\_NC*, and *Flag\_SC*, and they are initialized to 0 at the start. *Flag\_ET* is controlled by the *FIBC\_DT* and *NIBC\_DT* models to skip subsequent mode(s). To activate intra, normal IBC, or PLT modes, *Flag\_ET* should equal zero as shown in Fig. 8. The *CL\_DT* model updates *Flag\_NC* or *Flag\_SC* in accordance with CU type. *Flag\_NC* or *Flag\_SC* are updated to 1 for NC or SC, respectively. To check intra mode or the SCC tools, the flags should be checked first. For instance, normal IBC is performed under the condition that the CU size must be less than  $32 \times 32$  and *Flag\_ET* and *Flag\_SC* should be 0 and 1, respectively, as seen in Fig. 8.

After finishing the test of all modes and detecting the best mode, the encoder will perform a check to determine whether to terminate splitting of the current CU or not. If the CU size is  $8 \times 8$  or the *LR* value is smaller than a pre-defined TH, the encoder stops the partitioning procedure and get the best size. Otherwise, the CU moves to the higher depth level, then all modes are re-checked again as shown in Fig.8.

In the followings, the details about the training features will be discussed. Then, the training and implementation procedures of the decision tress will be explained.

### A. FEATURE EXTRACTION

Feature selection is key to obtaining well-trained models to improve the prediction accuracy. The methodology of how

the features are extracted from the CUs was explained before in sub-section II-B. As mentioned before, the blocks with limited colors are mostly being encoded by the SCC tools, and CUs that have major colors are efficiently encoded by the conventional intra mode. Thus, the number of distinct colors ( $D_c$ ) inside each CU is estimated as a feature. Natural videos contain an abundance of smooth regions. In other words, the luminance values are very close to each other in contrast to SC areas [34]. Therefore, *LR* (3) is considered as a feature. Horizontal activity  $H_{act}$  and vertical activity  $V_{act}$  are calculated to be used in the conventional SCM-6 for  $8 \times 8$  CUs. They are applied to normal IBC to determine if the search will be done in 1-D or 2-D regions. These values are calculated as training features, and are defined as:

$$H_{act} = \sum_{y=0}^{N-1} \sum_{x=1}^{M-1} |L_{x,y} - L_{x-1,y}| \quad (4)$$

$$V_{act} = \sum_{x=0}^{M-1} \sum_{y=1}^{N-1} |L_{x,y} - L_{x,y-1}| \quad (5)$$

where M and N represent the width and height of the CU, respectively, and  $L_{x,y}$  is the luminance value at position (x,y). To represent the complexity of the CU, a Texture complexity  $T_{com}$  parameter was extracted as a feature.  $T_{com}$  is estimated as explained in [35] as follows:

$$T_{com} = \frac{1}{M \times N} \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} (L_{x,y} - \overline{L_{x,y}})^2 \quad (6a)$$

$$\overline{L_{x,y}} = \frac{1}{8} \times \begin{bmatrix} L_{x-1,y-1} + L_{x-1,y} \\ +L_{x-1,y+1} + L_{x,y-1} \\ +L_{x,y+1} + L_{x+1,y-1} \\ +L_{x+1,y} + L_{x+1,y+1} \end{bmatrix} \quad (6b)$$

where  $\overline{L_{x,y}}$  is the mean luminance value of the 8 neighboring pixels around a pixel at  $(x, y)$ . SC blocks have sharp edges compared with NC blocks. For that, the edge complexity is measured for each pixel of the CU. The Sobel operator method is used to capture the edge information due to its simplicity. The operator components were measured in the vertical and horizontal directions in addition to diagonal directions at  $45^\circ$  and  $135^\circ$ . The parameter that represents the complexity of edges by using the Sobel operator components is denoted as  $E_{com}$ , and it is used as a feature.  $E_{com}$  was explained in detail in [35]. Fig. 9 illustrates the percentages of NC and SC CUs versus  $E_{com}$ , where these statistics were collected from data samples for  $8 \times 8$  CUs. From this figure, we can determine that as  $E_{com}$  increases, the number of SC blocks make up the majority of the CUs. For instance, if  $E_{com}$  is larger than 10K, the percentage of SC CUs is 88.45%, which means that most of the CUs that have sharp edges are encoded by IBC or PLT mode.

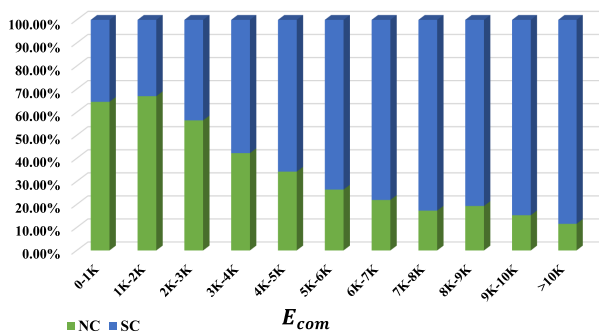


FIGURE 9. The percentages of NC and SC CUs versus  $E_{com}$  for  $8 \times 8$  CUs.

As mentioned before, obtaining the type of Parent CU improves the prediction accuracy of the current CU type. As a consequence, the encoder can run dedicated mode(s) for each type while remaining mode(s) will be skipped. Therefore, the flags that define the type of each Parent CU are collected to be included in the training features.  $P_{flag}$  denotes the Parent CU type flag,  $P_{flag} \in \{0, 1\}$ , 0 for intra mode, 1 for SCC modes.

For  $2N \times 2N$  fast IBC, there are flags that were implemented to give an indication when the RD cost of the  $2N \times 2N$  fast IBC mode is chosen as the best RD cost so far during the mode decision process, and that is because the conventional SCM sets its own IBC flag regardless whether  $2N \times 2N$  fast IBC, normal IBC, or Skip/Merge mode is selected as the optimum mode. To differentiate the case of  $2N \times 2N$  fast IBC as the optimum mode among others IBC modes, the  $FIBC\_flag$  was created for that,  $FIBC\_flag \in \{0, 1\}$ . A value of 1 means that the RD cost corresponding to the test of  $2N \times 2N$  fast IBC is the optimum cost. For all other cases,  $FIBC\_flag$  equals 0.  $J_{fbc}$  denotes the RD cost when  $FIBC\_flag$  equals 1, and is taken

as a feature for the training. By using the same procedures for normal IBC,  $J_{nibc}$  is the optimum RD cost when the best cost comes from the normal IBC search, and  $J_{nibc}$  values are collected to be used as a feature. In conclusion, the features that are used to train DTM models are  $D_c, LR, H_{act}, V_{act}, T_{com}, E_{com}, P_{flag}, J_{fbc}$ , and  $J_{nibc}$ .

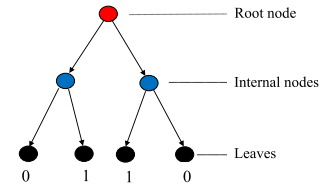


FIGURE 10. DT example.

### 1) TRAINING AND IMPLEMENTATION

A DT [36] is an ML technique, which is categorized as a supervised learning method. It is used to solve classification and regression problems. The DT technique is used in this work to build the predictive models of the DTM technique. The reason for choosing the DT technique is due to its simplicity during the training and implementation phases since it comes with a low complexity. Fig. 10 shows an example of a DT, which consists of a root node, internal nodes, and leaves. The non-leaf nodes perform a test on a feature or attribute. Then, these nodes are split into two internal nodes or leaf nodes. The leaf nodes output a class label, i.e., 0 or 1 as a prediction. One advantage of this technique is that it can be easily converted to If-else statements, which makes the software and hardware implementations more comfortable. As mentioned before, the DTM technique has three predictive models, in which two of them are used for skipping modes, while the third is used to predict whether the CUs are NC or SC. Fig. 11 illustrates the complete flow of the DTM technique, from the training sequences used for the training to obtaining the compressed files of the test sequences. In this figure, the conventional version of the HEVC+SCM software, which is HM-16.7+SCM-6, receives the training sequence at the beginning. After that, the features of the CUs and the class labels are extracted to formulate the feature vectors files, since each row in the feature vector file contains a feature vector and class label for a CU. These files are trained using an ML tool, which generates DTs as tree-like structures, which are converted to If-else statements to be incorporated into the HEVC+SCM version. From this, we have a DT-based HEVC+SCM version that includes the trained models. Eventually, the encoder can take test sequences and compress them faster than the conventional version.

The DTs in our work were created by using the Orange tool [37] version 3.32, which is a well-known open source software for ML, data-mining, and data analysis. It provides better visualization for the data since it uses widgets to connect the components through a Graphic User Interface (GUI).



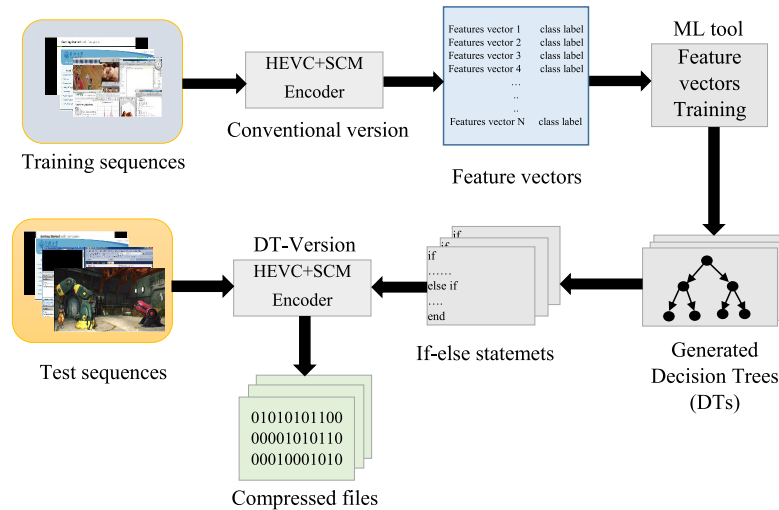


FIGURE 11. Work flow of the proposed DTM technique.

For each predictive model, the selected trained features should reflect the characteristics of its function. For *FIBC\_DT*, the outcomes from the model determine if 2N×2N fast IBC mode is sufficient to encode the CU and if it is unnecessary to check subsequent modes. Early termination decisions can be well-characterized by using the information of RD costs since a small RD cost indicates that the checked mode may be sufficient. Moreover, the CU structure affects the resultant RD cost, where complex-structure CUs are more likely to be encoded with a larger cost compared with the CUs with small details. Thus,  $J_{fIBC}$  and the features that reflect the CU structure are used to form the feature vectors files to train the *FIBC\_DT* model, where  $f_{fIBC}$  denotes the feature vector that is used for *FIBC\_DT* and is represented as:

$$f_{fIBC} = (J_{fIBC}, D_c, LR, T_{com}, E_{com}, H_{act}, V_{act}) \quad (7)$$

The training was done for each QP since the RD cost (1) is a function of a QP. The DTs were designed for the 8 × 8, 16 × 16, and 32 × 32 CUs. Table 4 shows the information of the DTs for the *FIBC\_DT* model.

TABLE 4. Information of the DTs for *FIBC\_DT*.

QP	CU size	Number of samples	Prediction accuracy (%)
QP22	8x8	1,990,262	87.8
	16x16	1,080,526	90.4
	32x32	147,296	91
QP27	8x8	1,822,920	87
	16x16	1,034,458	89.5
	32x32	144,488	90.7
QP32	8x8	1,246,264	88.2
	16x16	819,978	84.3
	32x32	114,464	90.6
QP37	8x8	996,430	87.3
	16x16	743,508	71.8
	32x32	149,314	85.6

From Table 4, the number of samples were selected from the data samples, 50% of the samples come from CUs that

have  $J_{fIBC}$  as the best cost with class labels of 1, while the other 50% come from CUs which do not take  $J_{fIBC}$  as the optimum cost with class labels of 0, and this concept is utilized for all DTs. Most of designed DTs achieve training accuracy more than 85%, which reflects the suitability of the trained features. The leaf nodes of the trees represent the *Flag\_ET* which was described before. The function of *NIBC\_DT* block is similar to *FIBC\_DT* but  $J_{nIBC}$  is used instead of  $J_{fIBC}$ . The feature vector of *NIBC\_DT* is called  $f_{nIBC}$  and is represented as:

$$f_{nIBC} = (J_{nIBC}, D_c, LR, T_{com}, E_{com}, H_{act}, V_{act}) \quad (8)$$

The training was done for 8 × 8 and 16 × 16 CUs at each QP. Table 5 shows the details about the designed DTs of the *NIBC\_DT* model. *Flag\_ET* is also set on the basis of the outcomes of these trees. We can see from Table 5 that the accuracies vary from 79.6% to 92.3%.

TABLE 5. Information of the DTs for *NIBC\_DT*.

QP	CU size	Number of samples	Prediction accuracy (%)
QP22	8x8	640,378	92.3
	16x16	512,540	79.6
QP27	8x8	570,576	91.5
	16x16	541,820	85.3
QP32	8x8	415,352	88.9
	16x16	517,572	85.1
QP37	8x8	345,632	78.9
	16x16	498,556	82.8

When it comes to *CL\_DT*, this model is responsible for categorizing the CUs into NC and SC. Therefore, the selected features should reflect the characteristics of the CUs types to enable the model to give an accurate prediction. As discussed before, the correlation between the type of CU and its Parent CU can be used to enhance the efficiency of the type predictors. Thus,  $P_{flag}$  and the features that reflect the CU structure form the feature vectors. A feature vector used in the training

of the  $CL\_DT$  model is called  $f_{cl}$ , and is defined as:

$$f_{cl} = (P_{flag}, D_c, LR, T_{com}, E_{com}, H_{act}, V_{act}) \quad (9)$$

The trained data was collected for QPs of 22, 27, 32, and 37 since the features do not include cost information. Table 6 shows the training information of data samples to generate the DTs of the  $CL\_DT$ . In Table 6, there are three trees that can be seen, each one acts as a classifier for a certain CU size. The accuracies are between 81.7% and 87.6%, which means that the classifier can predict the type of most CUs accurately. Each leaf has two cases, 0 and 1. When the outcome is 0, it indicates that the CU type is NC, so  $Flag\_NC$  will be set to 1, consequently, the SCC tool will be deactivated. Otherwise, the  $Flag\_SC$  will be set to 1 and the intra mode prediction will be discarded.

TABLE 6. Information of the DTs for  $CL\_DT$ .

CU size	No of samples	Prediction accuracy (%)
8x8	13,808,978	81.7
16X16	5,91,7914	85.2
32X32	1,668,808	87.6

#### IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

The performance of the proposed framework is evaluated from different aspects. The impact of each technique is analyzed, and the LR value for EST is suggested first. Then, the effectiveness of the early termination models in DTM is discussed. Following that, the results of the combination of DTM and EST will be presented. For further evaluation, simulations were conducted for a number of natural sequences that are recommended for the conventional HEVC. Finally, the proposed techniques are integrated with an existing approach to further speed up the encoding process, and the frameworks are compared with selected state-of-the-art approaches with respect to the conventional SCM-6.

All simulations were conducted using HEVC SC software model SCM-6. The QPs of 22, 27, 32, and 37 are used under the AI configuration profile.

##### A. THE IMPACT OF EACH PROPOSED TECHNIQUE

As discussed in Section II-B, homogeneous blocks have the chance to be encoded without splitting with negligible coding performance loss. In the EST technique, the homogeneity can be measured by checking the contrast of the luminance values inside the CU. Consequently, a decision can be made to determine if the splitting process may be terminated. In fact, choosing the LR value that can be considered as a threshold TH to judge the homogeneity depends on the requirements. A low TH value can preserve the loss but with minimal reduction in encoding time. On the other hand, a large TH value can achieve a larger reduction but at the expense of performance degradation. To investigate the impact of the different values of the LR on encoding time and the coding efficiency, simulations were conducted by using the first 10 frames of 6 selected sequences from Table 1,

which cover different categories. The test sequences are “WebBrowsing”, “SlideShow”, “Programming”, “MissionControlClip2”, “Robot”, and “Kimono1”.

The conventional SCM-6 version was run first. Then, it was modified to ignore the normal splitting process of  $16 \times 16$ ,  $32 \times 32$ , and  $64 \times 64$  CUs and if the LR value was smaller than a pre-defined TH. The modified version was evaluated many times at multiple thresholds (0, 16, 32, 48, and 64). This procedure was conducted for each CU size alone.

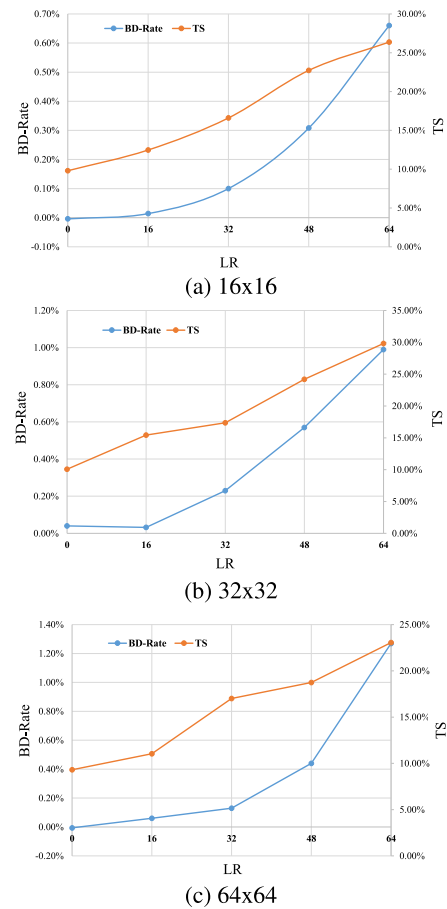


FIGURE 12. Impact on BR-Rate and TS against LR for each CU size. (a)  $16 \times 16$ , (b)  $32 \times 32$ , (c)  $64 \times 64$ .

Fig. 12 shows the impact of different LR values on the BD-Rate and TS. From the figure, we can notice that as the LR increases, the TS and BD-Rate increases. Fig. 12a shows the analysis for  $16 \times 16$  CUs. When the splitting process is ignored for uniform blocks where  $LR = 0$ , there is no change in BD-Rate approximately, which means that the performance is not affected and a 9.8% reduction in encoding time is achieved. When the LR equals 16, TS increases to 12.47% with negligible change in BD-Rate with respect to the previous TH, and the increase in BD-Rate is recorded as 0.01%. For a larger contrast, TS shows higher values at the expense of a higher BD-Rate increment. For example, in the case of  $TH = 32$ , the increase in BD-Rate is 0.1%, which is ten times that of the previous case with a 16.6% encoding

**TABLE 7.** BD-Rate and TS results of the three mode skipping models.

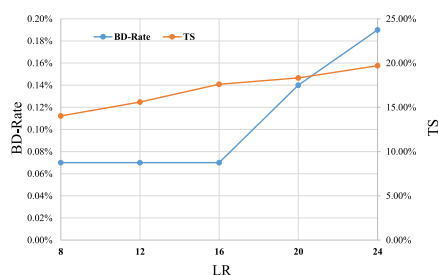
Sequences	Type	<i>FIBC_DT</i>		<i>NIBC_DT</i>		<i>CL_DT</i>	
		BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)
WebBrowsing	TGM	0.02	6.6	0.03	6.81	0.57	23.09
SlideShow	TGM	0.08	7.2	0	6.18	1.25	26.71
Map	TGM	-0.06	3.42	-0.06	4.66	0.93	21.15
BasketBallScreen	Mixed	0	4.05	0	4.81	0.7	26.05
Robot	A	-0.02	1.07	-0.01	2.02	0.79	50.66
Kimono1	CC	0	1.33	0	2.17	0.04	49.11
<b>Average</b>		<b>≈ 0</b>	<b>3.94</b>	<b>≈ 0</b>	<b>4.44</b>	<b>0.71</b>	<b>32.79</b>

**TABLE 8.** BD-Rate and TS results of the three DTM designs.

Sequences	Type	DTM-D1		DTM-D2		DTM-D3	
		BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)
WebBrowsing	TGM	0.33	21.76	0.2	22.79	0.13	26.66
SlideShow	TGM	2.19	29.2	2.32	31.01	1.92	34.35
Map	TGM	1.34	26.73	1.42	29.66	1.43	32.34
BasketBallScreen	Mixed	0.7	28.66	0.71	25.9	0.74	32.47
Robot	A	0.79	52.27	0.79	51.36	0.79	53.2
Kimono1	CC	0.05	50.73	0.05	49.92	0.05	50.06
<b>Average</b>		<b>0.9</b>	<b>34.89</b>	<b>0.91</b>	<b>35.1</b>	<b>0.84</b>	<b>38.18</b>

time reduction, and the difference in TS with respect to the previous point is not large. If the TH is 64, the encoder can work faster by 26.37% with a 0.66% increase in BD-Rate.

Figures 12b and 12c show similar patterns. From this, we can notice that the BD-Rate values show a dramatic change if the TH of EST is more than 16. To further analyze around the point of TH = 16, more simulations were performed in a similar way to the previous ones. However, the LR swapping is changed to be between 8 and 24 with 4 step increments. The simulations were conducted by adjusting TH for all CUs together. The results are shown in Fig. 13.

**FIGURE 13.** Impact on BD-Rate and TS against LR around TH = 16.

From Fig. 13, when the LR is smaller than 16, BD-Rate remains unchanged at 0.07%, while TS increases to 17.6%. Larger THs show significant increase in BD-Rate increment and a slow increase in time reduction. From the analysis, we can conclude that to determine a proper threshold, we must determine an acceptable performance loss. In our EST method, we need to maintain a high level of performance with an acceptable TS. Thus, the TH of EST is suggested to be 16 for all CUs on the basis of the experimental analysis.

As mentioned before, the DTM technique has three types of models. The *CL\_DT* model is used to predict the CU type, and the *FIBC\_DT* and *NIBC\_DT* models to skip

subsequent mode(s). These models reduce the time complexity of the encoder by determining if the subsequent mode(s) will be executed or not. To analyze the effectiveness of these models, Table 7 shows the BD-Rate and TS results of each mode skipping model with respect to the conventional SCM-6. Six videos from Table 1 were used for the assessment. From Table 7, it can be observed that the largest encoding time reduction is achieved by using *CL\_DT* by 32.79% and 0.71% increase in BD-Rate on average. “Robot” and “Kimono1” sequences show the largest encoding time reduction by 50.66% and 49.11%, respectively, while the BD-Rate increases by 0.79% and 0.04%, respectively. *FIBC\_DT* and *NIBC\_DT* models show no increase in BD-Rate approximately, whereas the TS results are 3.94% and 4.44%, respectively. “WebBrowsing” and “SideShow” sequences have the largest encoding time reduction at *FIBC\_DT* and *NIBC\_DT*, and that because these sequences have an abundance of repeated patterns. For further evaluation, we have implemented three designs for DTM, which are DTM-D1, DTM-D2, and DTM-D3. DTM-D3 includes all mode skipping models models, and *FIBC\_DT* and *NIBC\_DT* models are not included in DTM-D1 and DTM-D2, respectively. Table 8 tabulates the BD-Rate and TS results of the three DTM designs using the same test sequences appeared in Table 7. From Table 8, it can be observed that by incorporating the *FIBC\_DT* and *NIBC\_DT* models with the *CL\_DT* model, DTM-D3 provides a 38.18% encoding time reduction on average, which outperforms DTM-D1 and DTM-D2 by 3.29% and 3.08%, respectively. The increase in the BD-Rate for DTM-D3 is the lowest with 0.84% on average. DTM-D1 and DTM-D2 provide TS of 34.89% and 35.1% on average, respectively, while the increase in BD-rate is 0.9% and 0.91% on average, respectively.

We evaluate EST and DTM-D3 by considering the suggested TH for EST and by incorporating all predictive models

TABLE 9. Results of proposed techniques.

Sequences	Type	DTM-D3		EST		DTM-D3+EST	
		BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)
WebBrowsing	TGM	0.13	26.66	0.14	13.48	0.2	31.03
Map	TGM	1.43	32.34	0.03	8.8	1.59	37.18
Programming	TGM	0.83	23.28	0.24	7.75	1.15	28.41
Desktop	TGM	0.4	17.7	0.01	4.86	0.45	22.29
SlideShow	TGM	1.92	34.35	-0.16	40.09	2.12	55.19
FlyingGraphics	TGM	0.91	18.31	0.05	5.68	1.01	19.9
Console	TGM	0.73	17.97	-0.01	6.9	0.75	19.3
ChineseEditing	TGM	0.42	14.53	0	4.68	0.64	17.68
BasketballScreen	Mixed	0.74	32.47	0.01	10.37	0.78	35.06
MissionControlClip2	Mixed	1.23	39.02	-0.01	13.74	1.31	47.8
MissionControlClip3	Mixed	0.67	33.36	-0.04	10.56	0.74	39.19
Robot	A	0.79	53.2	0.05	12.31	0.86	57.17
Kimono	CC	0.05	50.06	0	8.78	0.07	57.4
<b>Average</b>		<b>0.78</b>	<b>30.25</b>	<b>0.02</b>	<b>11.39</b>	<b>0.89</b>	<b>35.96</b>

in DTM-D3. Table 9 shows the results of EST, DTM-D3, and EST integrated with DTM-D3 (DTM-D3+EST) using all recommended sequences appeared in Table 1. From this table, a 30.25% encoding time reduction on average was obtained by utilizing the DTM-D3 scheme while an increase in BD-Rate was only 0.78%. The largest reduction in DTM-D3 is seen for “Robot” and “Kimono1” with 53.2% and 50.06% with a 0.79% and 0.05% increase in BD-Rate, respectively. The reason is that most CUs in the “Robot” and “Kimono1” sequences are NC as described before, which can be efficiently predicted by  $CL\_DT$ . Consequently, the SCC tools are skipped. Mixed type sequences such as “BasketballScreen”, “MissionControl2”, and “MissionControl3” contain NC and SC CUs. In addition to that, there are many repeated patterns. The different characteristics in the mixed type can be exploited by predictive models to increase the encoding time reduction. Since, TS values were recorded as 32.47%, 39.02%, and 33.36%, respectively, while the BD-Rate was increased by 0.74%, 1.23%, and 0.67%, respectively. For the remaining sequences, which are of the TGM type, TS was between 14.53% and 34.35%, and an increase in BD-Rate was between 0.13% and 1.92%. This category includes mostly SC CUs, so fewer CUs discarded the IBC and PLT modes, which affects the time reduction because the SCC tools are a bottleneck of the SC encoding time. However,  $FIBC\_DT$  and  $NIBC\_DT$  could reduce the time further for sequences that have many repeated patterns such as “Map” and “SlideShow”, which achieved a 32.34% and 34.35% encoding time reduction with a 1.43% and 1.92% increase in BD-Rate, respectively. When it comes to the EST technique, the impact of this scheme can be viewed mostly on the sequences that are rich in homogeneous patterns such as “WebBrowsing”, “SlideShow”, and “MissionControlClip2”, where the encoding time was reduced to 13.84%, 40.09%, and 13.74%, respectively, while the BD-Rate was very close to that of the conventional SCM-6. As shown in Table 9, EST provides an 11.39% encoding time reduction on average with negligible performance loss since the increase in BD-Rate was 0.02%. Furthermore,

DTM-D3+EST achieved a 35.96% lesser encoding time on average while the BD-Rate increased by 0.89%. The highest time complexity reduction was observed for the “Robot” and “Kimono1” sequences with 57.17% and 57.4% with BD-Rate increment of 0.86% and 0.07%, respectively. The largest contribution for these sequences comes from the  $CL\_DT$  model, since these sequences contain fewer smooth CUs. It can be observed that “SlideShow” sequence reduces the encoding times in the two techniques, so on average, it obtains a huge encoding time reduction with 55.19% and an large increase in BD-Rate with 2.12%.

## B. FURTHER STUDIES OF THE PROPOSED FRAMEWORK

To further evaluate the proposed techniques, six test sequences from [38] were selected, which are recommended for the conventional HEVC test. Almost all the CUs found in these videos are NC, so fewer CUs can be encoded by IBC and PLT mode. The first 100 frames were encoded by using the conventional SCM-6. Then, three simulation cases were conducted. For the first case, the SCC tools were disabled, for the second, intra mode was skipped, while in the last case, the encoder was modified to incorporate the DTM-D3+EST techniques.

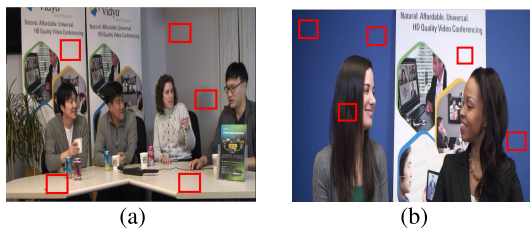
Table 10 shows the comparison results of the three cases with respect to the conventional SCM-6 in terms of BD-Rate and TS. As explained before, natural blocks are efficiently encoded by using the conventional intra mode rather than SCC tools. In Table 10, when SCC tools are bypassed, the sequences achieve a 66.6% reduction in encoding time with minimal degradation in coding performance with a BD-Rate increase of 1.93%. However, when the encoder utilizes the SCC tools only to encode the selected sequences, the performance is clearly affected, where the BD-Rate is increased by 34.67% on average and up to 43.96%. Finally, the proposed techniques give an average TS of 54.3% with low performance degradation since the increase in BD-Rate is only 0.74% on average. From these results, we can state that the proposed techniques can efficiently work with

**TABLE 10.** Results of three test cases for six natural sequences recommended for the conventional HEVC.

Sequences	SCC-disabled		Intra-disabled		DTM-D3+EST	
	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)
RaceHorses	-0.1814	68.72	39.65	-13.78	0.08	57.78
BQSquare	0.4796	69.39	19.67	8.76	0.42	44.99
BQmall	1.1154	70.61	34.66	-0.38	0.46	50.9
FourPeople	1.2536	63.93	38.95	-2.61	1.11	60.74
KristenAndSara	2.2763	58.52	43.96	0.58	0.96	60.29
BasketballDrillText	6.6819	68.46	31.13	-0.05	1.46	51.13
<b>Average</b>	<b>1.93</b>	<b>66.6</b>	<b>34.67</b>	<b>-1.24</b>	<b>0.74</b>	<b>54.30</b>

**TABLE 11.** Performance comparison with existing approaches with respect to the conventional SCM-6.

Sequences	Zhang [19]		Tsang [21]		Kuang [27]		RF [25]		DTM-D3+EST		DTM-D3+EST+Zhang[19]	
	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-Rate (%)	TS (%)	BD-R (%)	TS (%)
WebBrowsing	1.76	22.24	1.78	20.17	1.84	40.62	0.89	28.87	0.2	31.03	0.68	42.17
Map	0.52	24.05	1.6	13.82	1.72	29.55	0.84	25.65	1.59	37.18	2.2	47.29
Programming	0.59	20.4	1.44	16.94	1.76	37.92	1.81	24.98	1.15	28.41	1.64	40.11
Desktop	1.18	19.42	2.63	18.86	1.29	37.75	1.1	26.68	0.45	22.29	0.97	34.87
SlideShow	-0.02	34	1.26	42.42	4.44	37.47	4.42	25.84	2.12	55.19	2.58	62.5
FlyingGraphics	0.12	3.74	3.36	12.9	1.8	31.65	0.7	22.2	1.01	19.9	1.25	23.93
Console	0.9	17.69	3.65	13.7	1.16	39.97	0.86	27.21	0.75	19.3	1.35	35.54
chinese editing	0.58	29.92	1.28	14.3	0.68	32.5	1.2	29.74	0.64	17.68	1.06	38.75
BasketballScreen	0.78	25.75	1.3	18	1.83	39.5	1.74	26.62	0.78	35.06	1.39	47.86
MissionControl2	0.89	26.25	0.74	18.35	2.71	42.4	1.88	28.34	1.31	47.8	1.91	57.05
MissionControl3	0.49	25.23	1.49	17.25	1.86	39.82	1.47	30.03	0.74	39.19	1	48.16
Robot	0.2	6.98	0.15	15.94	1.79	53.7	0.87	9.91	0.86	57.17	0.89	57.73
Kimono	-0.01	1.38	0.01	8.25	0.72	45.5	0.05	42.4	0.07	57.4	0.06	60.08
<b>average</b>	<b>0.61</b>	<b>19.77</b>	<b>1.59</b>	<b>17.76</b>	<b>1.81</b>	<b>39.1</b>	<b>1.37</b>	<b>29.88</b>	<b>0.89</b>	<b>36.96</b>	<b>1.3</b>	<b>45.84</b>
CF	<b>32.41</b>		<b>11.17</b>		<b>21.6</b>		<b>21.81</b>		<b>41.52</b>		<b>35.26</b>	

**FIGURE 14.** Homogenous regions are highlighted by red squares (a) “FourPeople” sequence (b) “KristenAndSara” sequence.

camera-captured sequences as the conventional HEVC and limit the large encoding time brought by using conventional SCM encoders. Furthermore, the “FourPeople” and “KristenAndSara” sequences are rich in homogeneous patterns as shown in Fig. 14, where the smooth regions are highlighted by red squares. The compression of these types of sequences can be affected by the EST technique as they can achieve a larger time reduction as shown in Table 10, at 60.74% and 60.29% for “FourPeople” and “KristenAndSara” sequences, respectively, with minimal quality degradation.

### C. COMPARISONS WITH EXISTING APPROACHES

In this sub-section, the proposed framework is compared with other state-of-the-art approaches with respect to the conventional SCM-6. Four previous approaches were selected for the comparison; the frameworks in [19] and [21], can be

categorized as classical approaches, while the frameworks in [25] and [27] are ML-based techniques, which outperform other existing classical ML-based techniques as they have reported. The framework in [25] makes the SCM encoder faster by using the random forest method. The scheme in [27] employs DTs similar as our DTM-D3 technique. Furthermore, the approaches in [25] and [27] utilize fast mode skipping to reduce the encoding time, while in [19], the unnecessary modes are bypassed and fast IBC search was employed. The work in [21] adopts fast mode decisions, early pruning termination, and fast IBC search methods.

It should be noted that the approaches [19], [25], and [27] were implemented in SCM versions that differ from the SCM-6 used in this article, while SCM-6 was used in [21]. For a fair comparison, the source code of these schemes were re-implemented in SCM-6 to be simulated under the same conditions and test platform. The recommended test sequences shown in Table 1 are used for the evaluation. For further reduction in encoding time, the existing scheme in [19] has been integrated with the proposed DTM-D3+EST framework because it has the lowest performance degradation among the state-of-the-art works. In addition, it employs fast IBC search method since our framework does not include any fast IBC search algorithm.

Table 11 shows the comparison results of the proposed DTM-D3+EST framework with existing techniques as well as the results of combining the DTM-D3+EST framework

with the techniques in [19]. The proposed techniques can reduce the processing time by 36.96% on average, which outperforms [19], [21], and [27] that reduce the processing time complexity by 19.77%, 17.76%, and 29.88%, respectively. Also, the BD-Rate is increased by 0.89% on average, and that is better than [21], [25], and [27], since they increase the BD-Rate by 1.59%, 1.37%, and 1.81%, respectively. By integrating DTM-D3+EST with [19], a 45.84% encoding time reduction on average was achieved, which excels against the other schemes in Table 11, while the BD-Rate is increased by 1.3%. By applying the DTM-D3 technique, our framework ensures that at least one mode is skipped for the  $8 \times 8$ ,  $16 \times 16$ , and  $32 \times 32$  CUs during the mode decision process because the CUs perform dedicated mode(s) for each type when the CUs are classified by the  $CL\_DT$  model.

For each CU, intra mode is conducted only if the CU is classified as NC, normal IBC or normal IBC and PLT tools are conducted for SC CUs. Furthermore, the highest complexity reduction occurs if the  $FIBC\_DT$  model decides that the intra, normal IBC, and PLT modes need to be ignored. On the other hand, method [19] skips intra and normal IBC modes for static CUs only, and method [21] bypasses the SCC tools if the CU has a feature of vertical, horizontal, or 2-D pixel exactness. Techniques in [25] and [27] skip the modes in accordance with decisions made from the classification blocks placed before each mode. Therefore, in existing schemes, there is no guarantee that during the RDO process, one mode or more will be skipped.

To analyze the impact of the DTM-D3 trees, the “MissionControlClip2” sequence, shown in Fig. 1, is taken as an example. It contains NC and SC blocks. The encoding time is reduced for “MissionControlClip2” by 57.05% and 47.8% and the BD-Rate is increased by 1.91% and 1.31% if it is compressed by DTM-D3+EST+Zhang [19] and DTM-D3+EST, respectively, which is better than the other approaches. To further investigate the effectiveness of the  $CL\_DT$  DTs in the DTM technique, the results of the “Robot” and “Kimono1” sequences in Table 11 are taken as examples. Because most CUs in the “Robot” and “Kimono1” sequences are NC, the conventional intra is the most suitable mode to encode this type as discussed before. As seen in Table 11, the time reduction of these videos are 57.17% and 57.40% for DTM-D3+EST and 57.73% and 60.08% for DTM-D3+EST+Zhang [19], respectively, with a negligible increase in BD-Rate, which indicates that the  $CL\_DT$  trees can predict the CUs type with high accuracy.

The TS values are better than other schemes, especially when they are compared with those of [19] and [21]. The TS of “Robot” and “Kimono1” are 6.98% and 1.38% for [19] and 15.94% and 8.25% for [21], respectively, with a negligible change in the BD-Rate. That occurs because the CUs features that should exist to ignore modes by [19] and [21] are rare in the “Robot” and “Kimono1” sequences. As explained before, the videos that have a lot of homogeneous regions can be further compressed by applying early pruning termination methods. Our EST technique and the fast

CU size decision algorithm in [21] utilize an early termination method to reduce the number of RDO computations, while schemes [19], [25], and [27] do not employ those types of methods. To show the effect when the EST method is applied, we can take the “SlideShow” sequence as an example, since this sequence has an abundance of smooth blocks. By using [21], a 42.42% encoding time reduction is seen with an increase of 1.26% in the BD-Rate. DTM-D3+EST and DTM-D3+EST+Zhang [19] achieve a TS of 55.19% and 62.5% with a BD-Rate increase of 2.12% and 2.58%, respectively, which is faster than the approaches in the literature.

The better performance is achieved by utilizing larger encoding time reduction and a smaller BD-Rate increment. Thus, a comparison factor ( $CF$ ) is adopted as in [39] to evaluate the performance.

$$CF = \frac{TS}{BD - Rate} \quad (10)$$

A better performance is represented by a higher  $CF$  value. The  $CF$  value of each technique is shown in Table 11. It can be observed that the largest  $CF$  value is for the DTM-D3+EST techniques with 41.52, while scheme [21] has the worst performance with 11.17. The integration of DTM-D3+EST with Zhang [19] has a  $CF$  of 35.26, which is the second best performance scheme. However, it is the fastest scheme among all the works shown in Table 11.

The processing time for feature extraction and decision determination is 1.86% on average. This additional time has been counted in all simulations. For instance, in Table 11, the reported average encoding time reduction of 36.96% for the proposed algorithms (DTM-D3+EST) includes the extra processing time for feature extraction and decision determination. Hence, the time reduction would be 38.82% Without this extra processing time.

The maximum extra memory required for the proposed framework is  $64 \times 64 \times 8$  bits, i.e. 32 Kbit only, to estimate the  $64 \times 64$  pixels block features, which is the largest size. Here, the pixel value is represented by 8 bits. This extra memory is quite small and can be neglected.

## V. CONCLUSION

In this paper, a fast framework is proposed to speed up the encoding process of SC encoders. The framework contains two techniques. The first called DTM was designed by using the DTs classification method. In DTM,  $FIBC\_DT$  DTs are located after the test of  $2N \times 2N$  fast IBC to determine if intra, normal IBC, and PLT modes are skipped for the CUs of size  $8 \times 8$ ,  $16 \times 16$ , and  $32 \times 32$  pixels. Also, there are DTs that resides before the PLT mode to determine if the normal IBC is sufficient for  $8 \times 8$  and  $16 \times 16$  CUs or if the PLT mode should also be conducted.  $CL\_DT$  decision trees classify the CUs that are smaller than  $64 \times 64$  into NC or SC blocks. NC CUs are checked by using intra mode only, while intra mode is skipped for SC CUs. To further speed up the framework, a fast CU size decision method is adopted that skips the splitting process if the LR inside the CU is

smaller than a pre-defined threshold TH. A TH is determined on the basis of the performance requirement. The framework was implemented in SCM-6 for assessment. It can reduce encoding time by 35.96% on average while the BD-Rate increases to 0.89%. 54.3% encoding time reduction on average is achieved by using the proposed framework to compress natural sequences recommended for traditional HEVC with only 0.74% increment in the BD-Rate. By integrating the framework with existing approach in [19], the time saving increased to 45.84% with a BD-Rate increment of 1.3%. Our proposed framework is compatible with the reference HEVC SC encoder. This helps the proposed framework to be widely used in multimedia communication technologies to increase the video processing speed.

## ACKNOWLEDGMENT

The authors would like to thank E-JUST Center and Egypt-Japan University of Science and Technology for the continuous support. They would also like to thank the authors from [19], [21], [25], and [27] for the provision of source codes.

## REFERENCES

- [1] 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.
- [2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
- [3] S. Liu, X. Xu, S. Lei, and K. Jou, "Overview of HEVC extensions on screen content coding," *APSIPA Trans. Signal Inf. Process.*, vol. 4, pp. 1–4, Feb. 2015.
- [4] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging HEVC screen content coding extension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 50–62, Jan. 2016.
- [5] W.-H. Peng, F. G. Walls, R. A. Cohen, J. Xu, J. Ostermann, A. MacInnis, and T. Lin, "Overview of screen content video coding: Technologies, standards, and beyond," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 393–408, Dec. 2016.
- [6] M. Budagavi and D.-K. Kwon, "Intra motion compensation and entropy coding improvements for HEVC screen content coding," in *Proc. Picture Coding Symp. (PCS)*, Dec. 2013, pp. 365–368.
- [7] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, "2-D dictionary based video coding for screen contents," in *Proc. Data Compress. Conf.*, Mar. 2014, pp. 43–52.
- [8] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Hash based fast local search for intra block copy (IntraBC) mode in HEVC screen content coding," in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA)*, Dec. 2015, pp. 396–400.
- [9] X. Xu, S. Liu, T.-D. Chuang, Y.-W. Huang, S.-M. Lei, K. Rapaka, C. Pang, V. Seregin, Y.-K. Wang, and M. Karczewicz, "Intra block copy in HEVC screen content coding extensions," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 409–419, Dec. 2016.
- [10] X. Xu, S. Liu, T.-D. Chuang, and S. Lei, "Block vector prediction for intra block copying in HEVC screen content coding," in *Proc. Data Compress. Conf.*, Apr. 2015, pp. 273–282.
- [11] L. Guo, W. Pu, F. Zou, J. Sole, M. Karczewicz, and R. Joshi, "Color palette for screen content coding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 5556–5560.
- [12] X. Xiu, Y. He, R. Joshi, M. Karczewicz, P. Onno, C. Gisquet, and G. Laroche, "Palette-based coding in the screen content coding extension of the HEVC standard," in *Proc. Data Compress. Conf.*, Apr. 2015, pp. 253–262.
- [13] W. Pu, M. Karczewicz, R. Joshi, V. Seregin, F. Zou, J. Sole, Y.-C. Sun, T.-D. Chuang, P. Lai, S. Liu, S.-T. Hsiang, J. Ye, and Y.-W. Huang, "Palette mode coding in HEVC screen content coding extension," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 420–432, Dec. 2016.
- [14] X. Li, R. Wang, W. Wang, Z. Wang, and S. Dong, "Fast motion estimation methods for HEVC," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast.*, Jun. 2014, pp. 1–4.
- [15] Z. Pan, Y. Zhang, S. Kwong, X. Wang, and L. Xu, "Early termination for TZSearch in HEVC motion estimation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 1389–1393.
- [16] E. Badry, A. Shalaby, and M. S. Sayed, "Fast fractional-pixel motion estimation using lagrangian-based error surface interpolation," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2017, pp. 151–155.
- [17] D.-K. Kwon and M. Budagavi, "Fast intra block copy (IntraBC) search for HEVC screen content coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 9–12.
- [18] M. Zhang, Y. Zhang, and H. Bai, "Fast CU splitting in HEVC intra coding for screen content coding," *IEICE Trans. Inf. Syst.*, vol. E98.D, no. 2, pp. 467–470, 2015.
- [19] H. Zhang, Q. Zhou, N. Shi, F. Yang, X. Feng, and Z. Ma, "Fast intra mode decision and block matching for HEVC screen content compression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 1377–1381.
- [20] J. Lei, D. Li, Z. Pan, Z. Sun, S. Kwong, and C. Hou, "Fast intra prediction based on content property analysis for low complexity HEVC-based screen content coding," *IEEE Trans. Broadcast.*, vol. 63, no. 1, pp. 48–58, Mar. 2017.
- [21] S.-H. Tsang, Y.-L. Chan, W. Kuang, and W.-C. Siu, "Reduced-complexity intra block copy (IntraBC) mode with early CU splitting and pruning for HEVC screen content coding," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 269–283, Feb. 2019.
- [22] C. Huang, Z. Peng, F. Chen, Q. Jiang, G. Jiang, and Q. Hu, "Efficient cu and pu decision based on neural network and gray level co-occurrence matrix for intra prediction of screen content coding," *IEEE Access*, vol. 6, pp. 46643–46655, 2018.
- [23] W. Kuang, S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Fast mode decision algorithm for HEVC screen content intra coding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2473–2477.
- [24] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Online-learning-based Bayesian decision rule for fast intra mode and CU partitioning algorithm in HEVC screen content coding," *IEEE Trans. Image Process.*, vol. 29, pp. 170–185, 2020.
- [25] S.-H. Tsang, Y.-L. Chan, and W. Kuang, "Mode skipping for HEVC screen content coding via random forest," *IEEE Trans. Multimedia*, vol. 21, no. 10, pp. 2433–2446, Oct. 2019.
- [26] H. Yang, L. Shen, and P. An, "An efficient intra coding algorithm based on statistical learning for screen content coding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2468–2472.
- [27] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Machine learning-based fast intra mode decision for HEVC screen content coding via decision trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1481–1496, May 2020.
- [28] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Fast HEVC encoding decisions using data mining," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 4, pp. 660–673, Apr. 2015.
- [29] S. Zhu and C. Zhang, "A fast algorithm of intra prediction modes pruning for hevc based on decision trees and a new three-step search," *Multimedia Tools Appl.*, vol. 76, no. 20, pp. 21707–21728, 2017.
- [30] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra coding of the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1792–1801, Dec. 2012.
- [31] *HEVC Test Model Version 16.7 Screen Content Model Version 6.0*. Accessed: May 5, 2020. [Online]. Available: [https://hevc.hhi.fraunhofer.de/svn/svn\\_HEVCSoftware/tags/HM-16.7+SCM-6.0/](https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.7+SCM-6.0/)
- [32] H.-P. Yu, R. Cohen, K. Rapaka, and J.-Z. Xu, *Common Test Conditions for Screen Content Coding*, document JCTVC-X1015, May 2016.
- [33] G. Bjontegaard, *Calculation of Average PSNE Differences Between R-D Curves*, document VCEG-M33, ITU-T VCEG, 2001.
- [34] M. Zhang, S. Wang, and B. Li, "Selective motion estimation strategy based on content classification for HEVC screen content coding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2463–2467.

- [35] 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 Trans. Circuits Syst. Video Technol.*, vol. 29, no. 1, pp. 144–155, Jan. 2019.
- [36] L. Rokash and O. Maimon, *Data Mining with Decision Trees Theory and Applications*, 2nd ed. Singapore: World Scientific, 2015.
- [37] *Orange Data Mining Tool*. Accessed: Jun. 10, 2020. [Online]. Available: <http://orange.biolab.si>
- [38] F. Bossen, *Common Test Conditions*, document JCTVC-H1100, Mar. 2012.
- [39] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Fast HEVC to SCC transcoder by early CU partitioning termination and decision tree-based flexible mode decision for intra-frame coding," *IEEE Access*, vol. 7, pp. 8773–8788, 2019.



**EMAD BADRY** (Graduate Student Member, IEEE) received the B.Sc. degree in electronics and communication engineering from Suez Canal University, Ismailia, Egypt, in 2010, and the M.Sc. degree from the Egypt-Japan University of Science and Technology (E-JUST), New Borg El Arab, Egypt, in 2018, where he is currently pursuing the Ph.D. degree with the Department of Electronics and Communications Engineering. His current research interests include video coding, machine learning, and digital circuits design.



**KOJI INOUE** (Member, IEEE) received the B.E. and M.E. degrees in computer science from the Kyushu Institute of Technology, Japan, in 1994 and 1996, respectively, and the Ph.D. degree from the Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan, in 2001. In 1999, he joined Halo LSI Design and Technology, Inc., NY, USA, as a Circuit Designer.

He is currently a Professor of advanced information technology at Kyushu University. His research interests include power-aware computing, high-performance computing, secure computer systems, 3D microprocessor architectures, multi/many-core architectures, nanophotonic computing, and superconductor computing.



**MOHAMMED SHARAF SAYED** (Member, IEEE) received the B.Sc. degree in electronics and communications engineering from Zagazig University, Zagazig, Egypt, in 1997, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Calgary, Calgary, Canada, in 2003 and 2008, respectively.

He is currently an Associate Professor with the Department of Electronics and Communications Engineering, Egypt-Japan University of Science and Technology, Egypt. He holds one U.S. patent and seven contributions to the ITU/MPEG video coding standards, two of them included in the H.264 standard's reference hardware model. He has authored or coauthored more than 90 international publications. He managed and participated in several national/international research and development projects. His research interests include video coding, digital systems design, system-on-chip, embedded vision systems, and wireless body area networks.

Dr. Sayed received several awards both in Canada and Egypt. He acted as a reviewer for several IEEE TRANSACTIONS and other journals.

...