Received November 27, 2018, accepted December 22, 2018, date of publication January 3, 2019, date of current version January 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2890720

Fast HEVC to SCC Transcoder by Early CU Partitioning Termination and Decision Tree-Based Flexible Mode Decision for Intra-Frame Coding

WEI KUANG, (Student Member, IEEE), YUI-LAM CHAN, (Member, IEEE), SIK-HO TSANG, (Member, IEEE), AND WAN-CHI SIU, (Life Fellow, IEEE)
Center for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong

Corresponding author: Yui-Lam Chan (enylchan@polyu.edu.hk)

This work was supported in part by the Center for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, through the Research Studentship, and in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant PolyU 152112/17E.

ABSTRACT Screen content coding (SCC) was developed to enhance High Efficiency Video Coding (HEVC) for encoding screen content videos. However, HEVC has dominated the market for many years, and it leaves many legacy screen content videos encoded by HEVC. Therefore, it is desired that the legacy screen content videos are migrated from HEVC to SCC to improve the coding efficiency. This paper presents a fast transcoding algorithm by analyzing various features from four categories. They are the features from the HEVC decoder, static features, dynamic features, and spatial features. First, the coding unit (CU) depth level collected from the HEVC decoder is utilized to early terminate the CU partition in SCC. Second, a flexible encoding structure is proposed to make early mode decisions with the help of various features. On the one hand, high decision accuracy is achieved because mode decision is considered from different aspects by utilizing features from more than one category. On the other hand, high computational complexity is reduced because the flexible structure considers the decision of each mode separately. The experimental results show that the proposed algorithm provides 51.24% and 54.65% re-encoding time reduction with 1.32% and 1.25% negligible Bjøntegaard delta bitrate loss for YUV 4:2:0 and YUV 4:4:4 screen content sequences using all-intra configuration, respectively.

INDEX TERMS Transcoding, screen content coding (SCC), High Efficiency Video Coding (HEVC), fast algorithm, machine learning.

I. INTRODUCTION

Screen content video is an emerging video category due to the rapid development of computer technologies, and it plays a key role in many applications such as online education, remote desktop, WIFI display, video conference with document sharing, etc. Unlike the traditional natural videos captured by cameras, screen content videos are captured from the display screens of various electronic devices, and they usually show a mixed content of text, graphics, and natural images. High Efficiency Video Coding (HEVC) [1] has gained great success in compressing camera-captured videos based on the characteristics of natural image blocks (NIBs), and it roughly doubles the coding efficiency of its predecessor H.264/Advanced Video Coding (AVC) [2]. However, the special characteristics of screen content blocks (SCBs) such as sharp edges, repeated patterns and limited colors make HEVC inefficient for encoding screen content sequences. With the urgent demand for efficient screen content compression technologies, the Joint Collaborative Team on Video Coding (JCT-VC) developed the Screen Content Coding (SCC) extension [3] since 2014 and finalized it in 2016.

In the standardization of SCC, two important coding modes, intra block copy (IBC) mode [4], [5] and palette (PLT) mode [6], [7] have been included on top of the Intra mode of HEVC. IBC mode was designed as an ''intra version'' of the inter motion estimation. For a typical screen content video, there are many repeated patterns within the same frame. Therefore, IBC mode allows the current Coding Unit (CU) to search block vectors (BVs) in the reconstructed areas of the current frame. PLT mode was designed to take care of

CUs with limited colors, and it allows a CU to be encoded by several representative colors.

By adding coding tools specially designed for SCBs, SCC significantly improves the coding efficiency and it is expected to enhance HEVC to compress screen content videos. For example, the SCC reference software Screen Content Model version 4.0 (SCM-4.0) achieves over 50% Bjøntegaard delta bitrate (BDBR) [8] compared with the HEVC reference software HM-16.4 for typical screen content sequences [3]. However, the full adoption of SCC may take several years while HEVC is still a widely used video compression standard. The motivations of a HEVC to SCC transcoder are twofold. First, there are a significant number of legacy screen content videos encoded by HEVC, and it is necessary to convert them from HEVC to SCC to achieve low-cost storage. Second, a transcoder is desirable to alleviate the backbone traffic between the screen content center and cloud edges, where bandwidth resources are very expensive and therefore high-performance compression is demanded to break through the network bottleneck. When the screen content center uploads videos to cloud edges, it is necessary to convert HEVC bitstreams into SCC bitstreams with higher compression ratio for screen content videos. When users download videos from the cloud edges, they can either use a device with a SCC decoder or let the cloud edges convert the bitstreams back to HEVC.

In the literature, transcoding techniques can be divided into two categories: homogeneous transcoding and heterogeneous transcoding. Homogeneous transcoding refers to the conversion within the same format to meet a new functionality, such as different bit rates [9], different frame rates [10]–[12], different spatial resolutions [13], or even the insertion of new information such as watermarking [14] and error resilience layers [15], [16]. Heterogeneous transcoding refers to the bitstream conversion between different formats, and the HEVC to SCC transcoding belongs to this category. For heterogeneous transcoding, it can always be done by using a conventional brute-force transcoder (CBFT), which contains an original decoder and an original encoder. By taking HEVC to SCC transcoding as an example, CBFT decodes the incoming HEVC bitstream first, and then completely re-encodes the decoded video into a SCC bitstream. Although this approach can be applied to any heterogeneous transcoding tasks with high rate-distortion (RD) performance, it brings high computational complexity. Therefore, it is desired that the decoder side information can be collected to simplify the re-encoding process of CBFT, as shown in Fig. 1. Based on this idea, many fast transcoding algorithms have been proposed for different tasks, such as MPEG-2 to H.264 transcoding [17], [18], MPEG-2 to HEVC transcoding [19] and H.264 to HEVC transcoding [20]–[23]. These transcoders all focus on the fast CU partitioning decision because of different CU partitioning structures of the various standards. However, the fast HEVC to SCC transcoding is different from the previous transcoding problem due to the introduction of the new IBC and PLT modes. Therefore, new challenges are introduced in the HEVC to SCC transcoding problem.

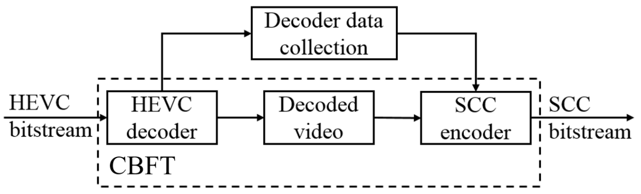


FIGURE 1. HEVC to SCC transcoder structure.

To reduce the computational complexity of a HEVC to SCC transcoder, one possible way is to use various fast encoding algorithms [24]–[26] to replace the original SCC encoder of CBFT in Fig. 1 by utilizing SCC encoder side information only. In [24], a fast CU size decision algorithm was proposed for stationary regions in screen content videos, where the collocated CU depth level and optimal mode are utilized to predict the current CU size. However, it is not suitable for screen content videos with many dynamic regions. In [25] and [26], algorithms were proposed to make both fast mode decision and fast CU size decision. In [25], it classifies CUs into NIBs and SCBs by content analysis. Only Intra mode is checked for NIBs. However, no fast mode decision is made for SCBs due to the low classification accuracy. Then, bit per pixel in the current CU and the depth information of neighbor CUs are used to make the fast CU size decision. In [26], Intra mode is checked for all CUs with 2N×2N prediction units (PUs). Then CUs are classified into partitioning CUs and non-partitioning CUs. For partitioning CUs, they directly go to the next depth level. For non-partitioning CUs, a similar strategy for fast mode decision as in [25] is adopted, where non-partitioning CUs are classified into SCBs and NIBs. Both PLT and IBC modes need to be checked for SCBs, while only Intra mode is checked for NIBs with N×N PUs at the depth level of 3. These fast algorithms all utilize the SCC encoder side information only, and they are not optimal when applied to the HEVC to SCC transcoding problem. On the one hand, the information from SCC encoder side contains noise due to the lossy encoding and decoding of HEVC, and it may lead to high RD performance loss. On the other hand, it is desired that the information from the HEVC decoder side can be utilized to improve the decision accuracy.

In the literature, there is only one paper [27] studying the fast transcoding scheme of HEVC to SCC, and it was designed for screen content videos in YUV 4:4:4 format. First, it directly maps the optimal CU size from HEVC to SCC and classifies CUs into non-partitioning CUs and partitioning CUs. For partitioning CUs, it utilizes a CU type classifier to further classify them into SCBs and NIBs like other fast mode decision algorithms [25], [26]. SCBs check both IBC and PLT modes before going to the next depth level, while NIBs directly go to the next depth level. For non-partitioning CUs, only Intra mode is checked and then

CU partitions are terminated. Besides, some thresholds are set to skip remaining modes and CU partitions if the bit cost of a CU is small. However, it has two drawbacks. First, it only utilizes some static features describing the current CU content to perform classifications while ignoring the information from the intermediate encoding stage and neighbor CUs. Therefore, the prediction accuracy is not high, and all mode candidates need to be checked for 8×8 CUs. Second, it treats the decision for IBC and PLT modes the same like other fast mode decision algorithms [25], [26], where both IBC and PLT modes are checked for SCBs. In fact, a SCB will only selects one mode from IBC and PLT modes, and higher re-encoding time reduction can be provided if mode candidates are further reduced for SCBs.

In this paper, we propose a fast HEVC to SCC transcoder (FHST) by utilizing various features from four categories. They are (a) features from the HEVC decoder, (b) static features describing CU content characteristics, (c) dynamic features extracted in the intermediate coding stage of SCC, and (d) spatial features from the sub-CUs and neighbor CUs. First, the optimal CU depth level collected from the HEVC decoder is utilized to early terminate CU partitions in SCC by using statistical analysis. Then, we propose a flexible mode decision structure to simplify the mode decision process. For each mode, a decision tree (DT) based classifier is inserted right before checking the target mode, and the most updated dynamic features are utilized to decide whether to check the mode or not. Therefore, IBC and PLT modes are no longer treated equally, and it allows the case that only one mode is checked for a SCB. Then, for a CU arrived at its last depth level, another DT-based classifier is trained by utilizing spatial features, and the decision of a mode is decided by a voting strategy. The differences between our contributions and the related schemes can be summarized as: 1) The works in [24]–[26] only utilize features from categories of (b), and the work in [27] uses features from categories of (a) and (b). However, we design multiple mode decision models by utilizing features from all the four categories to improve the decision accuracy. (2) The mode decision algorithms in [25]–[27] always treat the decision of IBC and PLT modes the same by performing CU type classification, and they are both checked for SCBs. On the contrary, the proposed flexible mode decision structure inserts a mode decision model before checking each mode, and many SCBs only check one mode of IBC or PLT, which leads to larger encoding time reduction. Considering that there are many existing screen content videos encoded by HEVC in YUV 4:2:0 format, we firstly propose FHST for YUV 4:2:0 screen content videos, and then extend the proposed FHST to support YUV 4:4:4 screen content videos. The source code of the proposed FHST can be found in our website http://www.eie.polyu.edu.hk/~ylchan/research/FHST/.

The rest of this paper is organized as follows. Section II presents the review on HEVC to SCC transcoding. Section III details the proposed FHST. The experimental results are presented in Section IV to verify the performance of the proposed algorithm. Finally, Section V concludes this paper.

II. REVIEW ON HEVC TO SCC TRANSCODING

A. DATA AVAILABLE FROM THE HEVC DECODER

When a HEVC encoder encodes a CU at the depth level of d, where $d \in \{0, 1, 2, 3\}$, it calculates the residual block R between the predicted CU, CU_{Pred}, and the original CU, CU_{Orig},

$$R = CU_{Orig} - CU_{Pred} \tag{1}$$

After transformation and quantization, the quantized transform coefficients C are obtained

$$C = (DRD^T) \otimes S_f \oslash Q \tag{2}$$

where D is the transformation matrix, S_f is the forward scaling matrix, Q is the quantization matrix, $\otimes$ is the element wise multiplication operator, and $\oslash$ is the element-wise division operator. Finally, the HEVC encoder signals the quantized transform coefficients C to represent the CU.

In the HEVC decoder side, the quantized transform coefficients C are obtained for each CU by decoding the HEVC bitstream. After the corresponding inverse transformation and dequantization processes, the reconstructed residual block R' is obtained as

$$R' = D^T(C \otimes Q \otimes S_i)D \tag{3}$$

where S_i is the inverse scaling matrix. Finally, a reconstructed CU, CU_{Reco}, is represented as

$$CU_{Reco} = R' + CU_{Pred} \tag{4}$$

To simplify the re-encoding process of SCC, the data from the HEVC decoder side can be utilized, such as the optimal depth level d, transform coefficients C and the reconstructed residual block R', as illustrated in Fig. 2.

B. REVIEW ON SCC ENCODER

As an extension of HEVC, SCC inherits the Coding Tree Unit (CTU) hierarchical partitioning structure from HEVC, as shown in Fig. 2(a). This structure lets an encoder select optimal CU sizes according to content adaptively. Starting from the size of 64×64 pixels at the depth level of 0, a CTU can be partitioned into 4 CUs of 32×32 pixels at the depth level of 1, and each CU continues partitioning with a minimum allowable size of 8×8 pixels at the depth level of 3.

In each CU, an exhaustive mode searching process is performed, as shown in Fig. 2(b). Besides the Intra mode inherited from HEVC, two additional coding modes, IBC and PLT modes, are included in SCC to improve the coding efficiency of screen content. When encoding a CU, IBC mode performs a ‘‘intra version’’ of motion estimation. It searches predictors in the reconstructed areas of the current frame, and a BV is signaled to indicate the relative location of the best predictor. To achieve an optimal balance of coding efficiency and computational complexity, it contains three steps

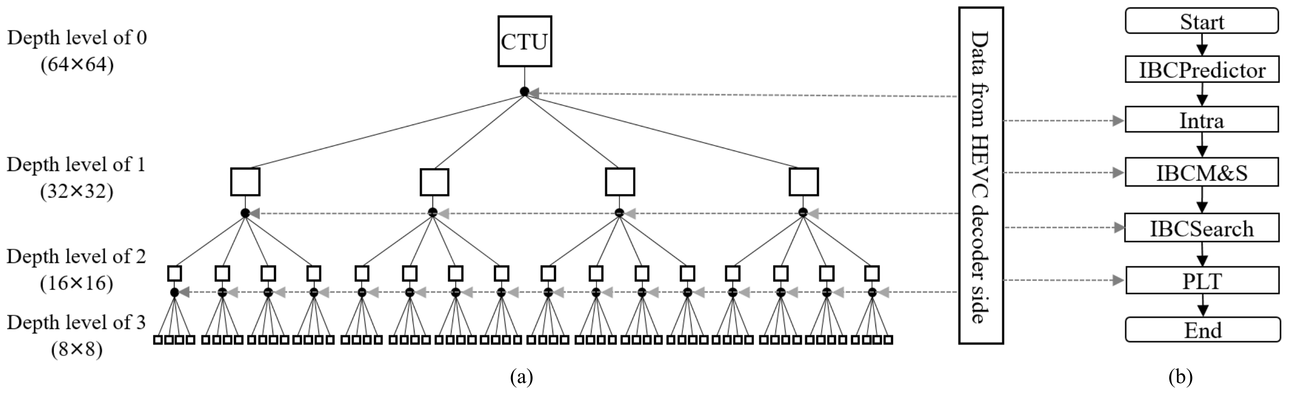


FIGURE 2. SCC re-encoding structure. (a) CTU hierarchical partitioning structure, and (b) Optimal mode searching structure.

with early termination conditions, which are IBCPredictor, IBCMerge&IBCSkip (IBCM&S) and IBCSearch. IBCPredictor is applied to CUs with sizes from 8×8 to 32×32, and it simply checks several BVs predicted from the last coded CUs and the neighbor CUs if they are encoded by IBC mode. If the distortion of checking IBCPredictor is zero, Intra mode will be skipped. IBCM&S is a "intra version" of the inter Skip and Merge modes in HEVC, and it is applied to CUs with sizes from 8×8 to 64×64. If IBCSkip is selected as the best mode so far, the optimal mode searching process is terminated. Otherwise, the remaining mode candidates need to be checked. IBCSearch performs motion estimation in the reconstructed area of the current frame, and it is applied to CUs with sizes from 8×8 to 16×16. PLT mode is designed for screen content with limited colors, and it is applied to CUs with sizes from 8×8 to 32×32. PLT mode selects several representative colors in a CU to form a palette, and then it compresses the CU by using an index map.

Similar to HEVC, the optimal CTU partitioning structure and its prediction modes are decided by performing the RD optimization (RDO) process. In a CU, RDO calculates a RD cost, $J(m)$, for each mode m to evaluate its performance

$$J(m) = Dist(m) + \lambda \cdot Rate(m) \tag{5}$$

where $Dist(m)$ represents the distortion between CU_{Orig} and CU_{Reco}, $Rate(m)$ denotes the total bits used to signal the CU when selecting the mode m, and λ is the Lagrange multiplier. By comparing the RD cost among different mode candidates and CU depth levels, the optimal mode in a CU is selected as the one with the smallest value of $J(m)$, while the optimal CTU partitioning structure is decided as the one with the smallest total value of $J(m)$.

III. PROPOSED FHST

To meet the challenge of computation-constrained applications, it is desired that the features from both the HEVC decoder and the SCC encoder are collected to simplify the re-encoding process. First, the features are utilized to early terminate the CTU partitions. Second, the features are also used to skip unnecessary mode candidates in a CU. Since the

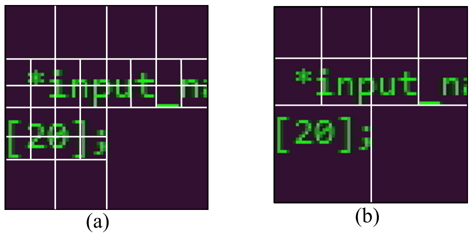


FIGURE 3. The partitioning structure of a CTU encoded by (a) HEVC and (b) SCC.

computational complexity of IBCPredictor is relatively low and our experiments show that it takes up only 2.04% of the total re-encoding time, we investigate the fast decisions for other modes such as Intra, IBCM&S, IBCSearch and PLT modes in our paper.

A. EARLY CU PARTITIONING TERMINATION

Although HEVC and SCC share the same CTU partitioning structure, the optimal CU size decided by HEVC may change during the transcoding process, and an example is shown in Fig. 3. It is observed that due to the adoption of the new coding modes, SCC allows inhomogeneous content to select larger CU sizes than HEVC. However, we also notice that most optimal CUs decided by HEVC do not continue partitioning in SCC. Therefore, the CU partitioning process in SCC can be early terminated by utilizing the decoder side information of HEVC.

To derive the early CU partitioning termination rule, we encoded and decoded 13 typical SCC test sequences [28] by HEVC reference software HM-16.12 [29] with quantization parameters (QPs) of 22, 27, 32, and 37 under All Intra (AI) configuration, and then the decoded sequences were re-encoded by SCC reference software HM-16.12+SCM-8.3 with the same QPs. Table 1 shows the average percentages of the further partitioned CUs from HEVC to SCC, where d_{HEVC} and d_{SCC} represent the CU depth level in HEVC and SCC, respectively. It is observed that for CUs with d_{HEVC} of 1 or 2, they rarely continue partitioning in SCC.

TABLE 1. Percentages of further partitioned CUs from HEVC to SCC.

d_{HEVC}	Partitioned to $d_{SCC}=1$ (%)	Partitioned to $d_{SCC}=2$ (%)	Partitioned to $d_{SCC}=3$ (%)
0	11.02	0.91	0.23
1		4.83	0.25
2			3.25

However, for CUs with d_{HEVC} of 0, 11.02% of them are partitioned to d_{SCC} of 1. Based on this observation, we set the early CU partitioning termination rule for different CU sizes adaptively by limiting the maximum CU depth level d_{SCC}^{max} allowed to be checked as

$$d_{SCC}^{max}=\begin{cases}1, & \text{if } d_{HEVC}=0\\ d_{HEVC}, & \text{otherwise.}\end{cases}\tag{6}$$

Therefore, for CUs with d_{HEVC} of 1 or 2, further partitions are not allowed in SCC. For CUs with d_{HEVC} of 0, FHST only allows them to be partitioned to d_{SCC} of 1, and then further partitions are terminated.

B. FLEXIBLE MODE DEICSION

We propose a flexible mode decision structure in our algorithm, where the decision of each mode is considered separately. The objective of the flexible mode decision technique is to design a decision model for each mode, so that it can assist the transcoder in making the decision of checking a mode or, on the contrary, skipping a mode. Therefore, in the re-encoding process, two classes are defined for a mode x, where $x \in$ {Intra, IBCM&S, IBCSearch, PLT}, i.e., checking the mode (ω_x) and skipping the mode ($\overline{\omega_x}$). By collecting features from both the HEVC decoder and the SCC encoder, the objective can be solved as a supervised classification task, and the model G is represented as

$$G(f_1, f_2, f_3, \ldots, f_n) \rightarrow \{\omega_x, \overline{\omega_x}\}\tag{7}$$

where f_i represents the features used to generate the model and $i = 1, .., n$. Therefore, the decision of each mode is made adaptively by inserting a model G before checking a mode.

1) ALGORITHM DESCRIPTION

In our paper, we implement a DT-based mode decision model right before checking a mode. Therefore, some dynamic features showing the intermediate coding information, such as the RD cost and IBC mode flag, can be employed to make mode decisions. Fig. 4 shows the DT-based x mode decision model of our proposed algorithm, where $x \in$ {Intra, IBCM&S, IBCSearch, PLT}. It can be seen from Fig. 4 that the x mode decision model contains two parts, which are the x mode classifier and the Spatial-Info classifier. The x mode classifier utilizes the features from the current CU to make the decision of a mode x, including features from both the HEVC decoder side and the SCC encoder side, and the class label is set to ω_x if the outcome of the x mode classifier is 1. Besides, we find that the mode decision accuracy for CUs arrived at their last depth levels, i.e., d_{SCC}^{max}, is very important to reduce

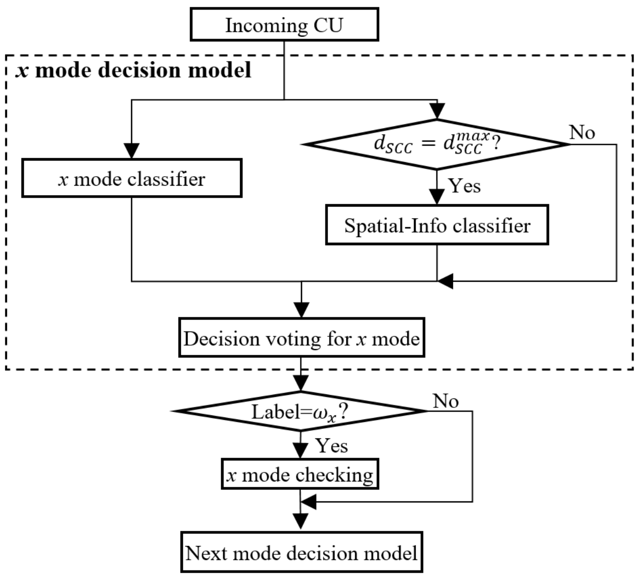


FIGURE 4. DT-based x mode decision model.

the RD performance loss. The reason is that if the optimal mode is skipped for a large CU, it can still find relatively good modes when it continues partitioning and arrives at its last depth level. To achieve a good trade-off between the computational complexity and coding efficiency, we additionally train a set of Spatial-Info classifiers for CUs arrived at their last depth levels. In our paper, we define CUs coded by Intra mode as NIBs and CUs coded by IBC or PLT mode as SCBs. The Spatial-Info classifier is trained by utilizing the spatial features to decide whether the current CU is a NIB or a SCB. If it is a SCB, the class label is set to ω_x for $x \in$ {IBCM&S, IBCSearch, PLT}. Otherwise, the class label is set to ω_x for $x \in$ {Intra}. After going through the classifiers, a decision voting strategy is adopted for making decisions. The label of the x mode decision model is set to ω_x if at least one classifier outputs ω_x. More discussions on the structure of the x mode decision model are provided in Section IV.C.

2) FEATURE SELECTION

To make the mode decision for a CU, features from 4 categories are considered: (a) features from the HEVC decoder, (b) static features describing CU content characteristics, (c) dynamic features extracted in the intermediate coding stage of SCC, and (d) spatial features from the sub-CUs and the neighbor CUs. Based on our prior knowledge on screen content videos, 16 features are selected, where features 1-10 are used to train the x mode classifier and features 11-16 are used to train the Spatial-Info classifier.

a: FEATURE SELECTION OF x MODE CLASSIFIER

i) FEATURES FROM THE HEVC DECODER

f_1: The average CU depth level of HEVC d_{HEVC}^{avg}, which is represented as

$$d_{HEVC}^{avg}=\frac{\sum_{d_{HEVC}} Area\,(d_{HEVC}) \times d_{HEVC}}{2\text{N} \times 2\text{N}}\tag{8}$$

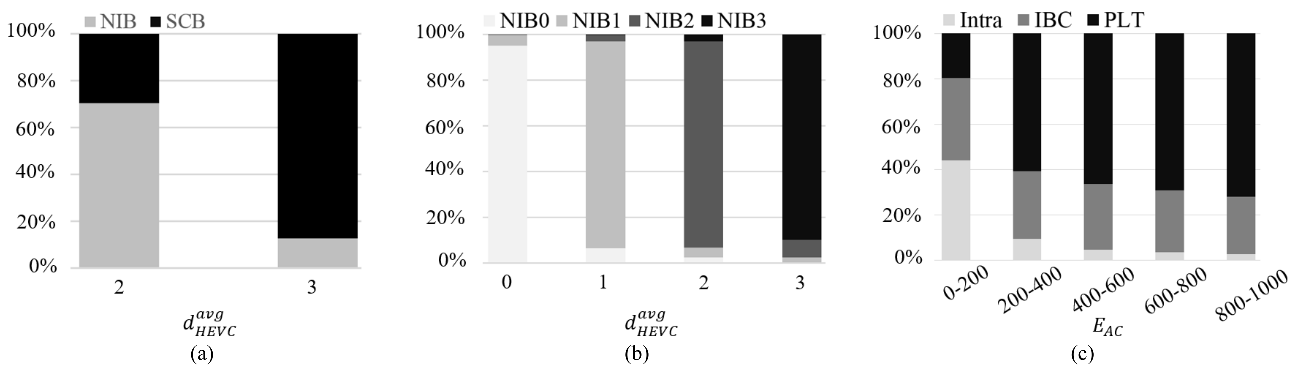


FIGURE 5. (a) NIB and SCB distribution over d_{HEVC}^{avg} for 16×16 CU size, (b) Optimal CU depth level distribution of NIBs over d_{HEVC}^{avg}, and (c) Mode distribution over E_{AC}.

where $Area\ (d_{HEVC})$ represents the area with d_{HEVC} in a CU, and 2N×2N represents the size of the CU. In HEVC, screen content is encoded by small CUs due to their inhomogeneity. Therefore, CUs with larger values of d_{HEVC}^{avg} are more likely to be SCBs. To verify this claim, 10000 16×16 NIBs and 10000 16×16 SCBs were randomly selected from the training set, and the distributions of NIBs and SCBs over d_{HEVC}^{avg} are given in Fig. 5(a). Since CUs with d_{HEVC}^{avg} of 0 and 1 are usually encoded as 64×64 or 32×32 CUs in SCC, Fig. 5(a) only shows the 16×16 CU type distributions over d_{HEVC}^{avg} of 2 and 3. It is observed that many CUs with d_{HEVC}^{avg} of 2 are NIBs while most CUs with d_{HEVC}^{avg} of 3 are SCBs. Besides, the CU depth level distribution of NIBs in SCC over d_{HEVC}^{avg} is investigated by randomly selecting 10000 NIBs at each depth level, and the results are shown in Fig. 5(b), where NIB0, NIB1, NIB2 and NIB3 denote the NIBs encoded at the depth levels of 0, 1, 2, and 3 in SCC, respectively. It is observed that that NIBs from HEVC are very likely to be encoded at the same depth levels in SCC. Therefore, Intra mode at other depth levels is very likely to be skipped for NIBs.

f_2: The AC coefficient energy E_{AC} of the quantized transform coefficients C, which is defined as the sum of square of the AC coefficients

$$E_{AC} = \sum_{c_{i,j}\in C, c_{i,j}\neq c_{0,0}} c_{i,j}^2 \tag{9}$$

where $c_{i,j}$ is a quantized transform coefficient with row index of i and column index of j in C, and $c_{0,0}$ is the DC coefficient. SCBs have many high frequency components and they contain higher values of E_{AC} than NIBs. Besides, we also notice that while many IBC coded CUs are smooth, PLT coded CUs are usually more complex, and they have even higher values of E_{AC} than IBC coded CUs. Statistics that support this claim are shown in Fig. 5(c), where 10000 Intra, 10000 IBC and 10000 PLT coded 16×16 CUs were randomly selected from the training set. Therefore, E_{AC} provides a chance to differentiate PLT mode from IBC mode.

f_3: The number of zeros in the residual block N_{ZR}, which is also adopted in the fast transcoding algorithm [27], and it

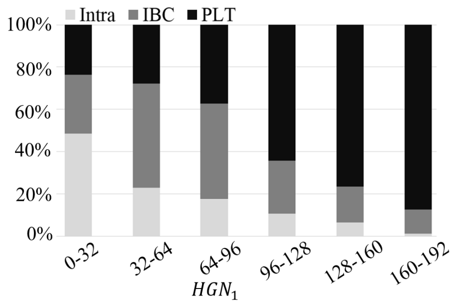


FIGURE 6. Mode distribution over HGN_1.

is defined as

$$N_{ZR} = \sum_{r_{i,j}\in R'} \delta(r_{i,j}, 0) \tag{10}$$

where $r_{i,j}$ is an element with row index of i and column index of j in the reconstructed residual block R', and Kronecker delta $\delta\left(r_{i,j}, 0\right)$ is represented as

$$\delta\left(a, b\right) = \begin{cases} 0, & \text{if } a \neq b \\ 1, & \text{if } a = b \end{cases} \tag{11}$$

Since SCBs have many uniform background pixels, they tend to have a larger value of N_{ZR}.

ii) STATIC FEATURES

$f_4 - f_7$: High gradient pixel number HGN_0, HGN_1, HGN_2, HGN_3. A pixel is defined as a high gradient pixel if the luminance difference of the current pixel $Y_{i,j}$ and one of the neighbor pixels $Y_{i\pm1,j}$, $Y_{i,j\pm1}$ is larger than a threshold TH_{HG}

$$\left|Y_{i,j} - Y_{i\pm1,j}\right| > TH_{HG}\ or\ |Y_{i,j} - Y_{i,j\pm1}| > TH_{HG} \tag{12}$$

where i and j denote the row and column indices of a pixel. To detect high gradient pixels with different strength, HGN_0, HGN_1, HGN_2, HGN_3 are counted with TH_{HG} of 8, 16, 32 and 64, respectively. Mode distribution over HGN_1 is shown in Fig. 6, where 10000 Intra, 10000 IBC and 10000 PLT coded 16×16 CUs were randomly selected from the training set. It is observed that SCBs have larger high

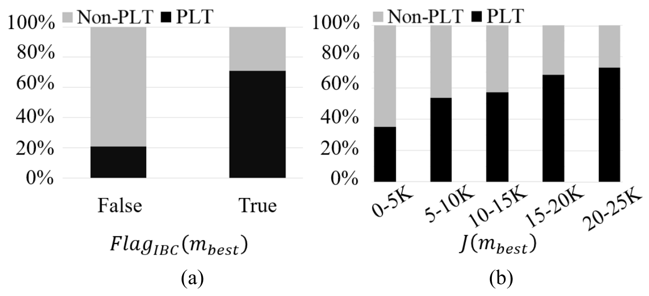


FIGURE 7. PLT coded CUs and Non-PLT mode coded CUs distribution over (a) $Flag_{IBC}$ (m_{best}) and (b) J (m_{best}).

gradient pixel number because they have many sharp edges. Besides, because PLT coded CUs are usually more complex, they also have larger high gradient pixel number than IBC coded CUs.

f_8 : Distinct color number N_{DC}, which is also used in fast SCC encoding algorithms [25], [26] and the fast transcoding algorithm [27]. It is calculated by counting the pixels with different luminance values. Since SCBs contain limited colors, they usually have smaller value of N_{DC} than NIBs.

iii) DYNAMIC FEATURES

$f_9 - f_{10}$: The RD cost and IBC flag of the best mode, J (m_{best}) and $Flag_{IBC}$ (m_{best}), respectively, before checking the target mode in the current CU. These two dynamic features reveal the intermediate coding stage before checking the target mode. J (m_{best}) shows the coding efficiency of a CU and $Flag_{IBC}$ (m_{best}) denotes if the CU is IBC coded before checking the target mode. If $Flag_{IBC}$ (m_{best}) is true, the current CU is more likely to be a SCB and it has a large chance to check the following IBC and PLT modes. Besides, if the value of J (m_{best}) is large before checking the target mode, the target mode is more likely to be checked. For example, if the target mode is PLT mode, it is more likely to be checked if $Flag_{IBC}$ (m_{best}) is true and the value of J (m_{best}) is large. To support this, 10000 PLT coded 16×16 CUs and 10000 Non-PLT coded 16×16 CUs were randomly selected from the training set, and mode distributions over $Flag_{IBC}$ (m_{best}), and J (m_{best}) before checking PLT mode are shown in Fig. 7.

b: FEATURE SELECTION OF SPATIAL-INFO CLASSIFIER

i) SPATIAL FEATURES

f_{11}: The neighbor SCB number, *SCBNum*, by counting the top and left neighbor CUs. *SCBNum* is denoted by

$$SCBNum = \delta(m_L, IBC) + \delta(m_L, PLT) + \delta(m_T, IBC) + \delta(m_T, PLT) \quad (13)$$

where m_L and m_T represent the optimal modes of the left and top CUs of the current CU, respectively. The current CU is more likely to be a SCB if it has SCB neighbors, and the evidence is shown in Fig. 8(a), where 10000 16×16 NIBs and 10000 16×16 SCBs were randomly selected from the training set.

f_{12}: The same background sub-CUs number, *BGCUNum*, by counting its four sub-CUs which have the same background color as the current CU. *BGCUNum* is defined as

$$BGCUNum = \sum\nolimits_{i=0}^{3} \delta(Y_{BC,d}, Y^i_{BC,d+1}) \quad (14)$$

where $Y_{BC,d}$ and $Y^i_{BC,d+1}$ are the background colors of the current CU at the depth level of d and its *i-th* sub-CU at the depth level of $d+1$, respectively. We define the background color in a CU/sub-CU as the luminance value with the highest occurrence frequency within the CU/sub-CU. If more sub-CUs have the same background color as the current CU, it is more likely to be a SCB. To support our claim, 10000 16×16 NIBs and 10000 16×16 SCBs were randomly selected, and their distributions over *BGCUNum* are shown in Fig. 8(b).

$f_{13} - f_{16}$: The high gradient pixel strength, HGS_0, HGS_1, HGS_2 and HGS_3, which are calculated by considering if the neighbor CUs from the left, right, top and bottom contain high gradient pixels with TH_{HG} of 8, 16, 32 and 64, respectively. Let us call a CU that contains high gradient pixels as a high gradient CU (HGCU). It is observed that if the current CU is a HGCU, the more neighbor HGCUs it has, the more likely it is a SCB. Otherwise, if the current CU is a non-HGCU, the more neighbor non-HGCUs it has, the more likely it is a NIB. We set the initial value of HGS_i $(i \in \{0, 1, 2, 3\})$ to 0. If the current CU is a HGCU, HGS_i is set to a non-negative value, and its absolute value is calculated by counting the number of HGCUs from the left, right, top and bottom neighbor CUs. Otherwise, if the current CU is a Non-HGCU, HGS_i is set to a non-positive value, and its absolute value is calculated by counting the number of Non-HGCUs from the left, right, top and bottom neighbor CUs. Therefore, as the value of HGS_i goes from small to large, the probability of the current CU being a SCB is increased, and statistics that give the evidence in this observation is shown in Fig. 8(c), where 10000 16×16 NIBs and 10000 16×16 SCBs were randomly selected.

c: TRAINING IMPLEMENTATION

Since there are numerous mode candidates in different CU sizes, it is difficult to manually select the optimal features and classification criterions to perform classification. In our paper, we select DTs as the classification model, and the optimal features with classification criterions are selected based on off-line training data. DTs come with low complexity in the testing phase, and they can be easily implemented in the transcoder as a set of "if-else" conditions. A DT is flowchart-like structure, and an example is shown in Fig. 9. In our case that decides whether a mode is checked or not, a CU with several features is input to the DT. Each non-leaf node runs a test on a feature, and each branch denotes an outcome of the test. After going through a series of test, the CU comes to a leaf node, and a class label of ω_x or $\overline{\omega_x}$ is assigned to it. Specifically, the class label is decided as the label of majority training samples in a leaf node, and the decision accuracy of a leaf node is denoted by the percentage of correctly classified samples in it. The DTs were trained in the Waikato

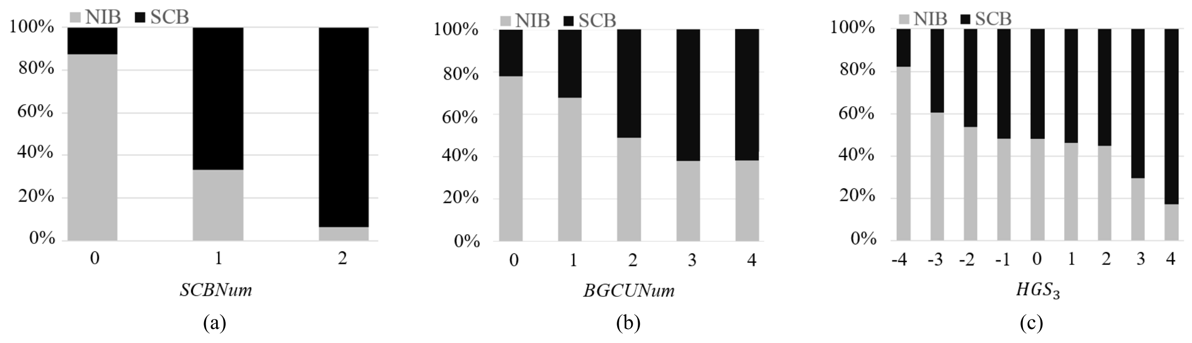


FIGURE 8. NIB and SCB distribution over (a) ***SCBNum***, (b) ***BGCUNum*** and (c) ***HGS_3***.

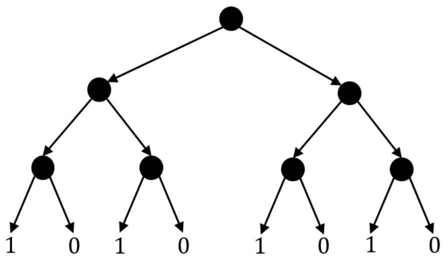


FIGURE 9. An example of a decision tree.

Environment for Knowledge Analysis (WEKA) [31] version 3.8 by using C4.5 algorithm [32], which adopts the gain ratio to select the best feature and the splitting criterion for each non-leaf node. The data impurity in a node t is measured by the entropy $En(t)$, and it is calculated as

$$En(t) = \sum\nolimits_{\omega} p(\omega) log_2 p(\omega) \tag{15}$$

where $p(\omega)$ is the percentage of the data belong to ω in the node, and $\omega \in \{\omega_x, \overline{\omega_x}\}$. To reduce data impurity, a feature f with a splitting criterion H is selected to split a node t to two child nodes t_k, $k \in \{L, R\}$. The data impurity reduction $Gain(f, H)$ is calculated as

$$Gain(f, H) = En(t) - \sum\nolimits_{k} \frac{N_k}{N_P} En(t_k) \tag{16}$$

where N_P and N_k are the number of samples in the parent node t and the child nodes t_k. Then the gain ratio is represented as

$$GainRatio(f, H) = \frac{Gain(f, H)}{SplitInfo(f, H)}. \tag{17}$$

where the normalization term $SplitInfo(f, H)$ is defined by

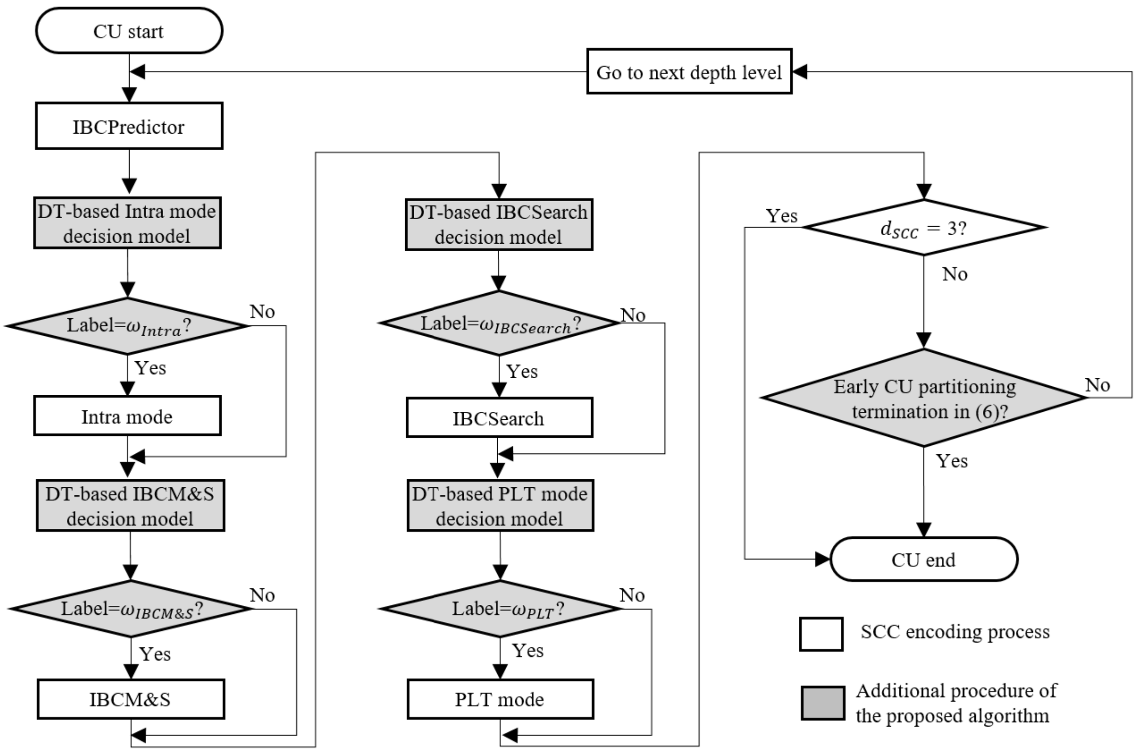
$$SplitInfo(f, H) = -\sum\nolimits_{k} \frac{N_k}{N_P} log_2 \frac{N_k}{N_P}. \tag{18}$$

The best feature and the splitting criterion in a node t are selected as the one by maximizing $GainRatio(f, H)$. A DT is trained node by node, and the node splitting is terminated when the number of training samples arrived at a node is less than or equal to 1% of the total training samples.

The training data for building DTs are selected from 5 sequences, which are "Console", "Desktop", "Map", "MissionControlClip2", and "Robot". To generate the training data, 10 frames were extracted from each sequence with equal time interval. Those frames were firstly encoded by HM-16.12 with QPs of 22, 27, 32 and 37, and then the decoded frames were re-encoded by HM-16.12+SCM-8.3 with the same QPs. When training the x mode classifier, the positive data come from CUs encoded by x mode, and the negative data are from CUs encoded by other modes. Therefore, the class label of x mode is ω_x if the outcome is 1. Otherwise, the class label is $\overline{\omega_x}$. To train the Spatial-Info classifier, the positive data are collected from SCBs while the negative data are collected from NIBs. Therefore, for SCBs, i.e., $x \in$ {IBCM&S, IBCSearch, PLT}, the class label is set to ω_x if the outcome is 1. For NIB, i.e., $x \in$ {Intra}, the class label is set to ω_x if the outcome is 0. To avoid the data imbalance problem [33] caused by more training samples in one class than the other, we set the numbers of positive and negative training data to be equal. To balance the coding efficiency and computational complexity, we set two confidence thresholds α and β in the Spatial-Info classifier and x mode classifier, respectively. If the accuracy of a decision made by the Spatial-Info classifier or x mode classifier is lower than the value of α or β, the class label for mode x is set to ω_x regardless of its outcome.

As SCC inherits the CTU hierarchical partitioning structure from HEVC, which supports 4 different CU sizes from 8×8 to 64×64, the classifiers for each mode was trained for CUs with different sizes, respectively.

As a summary, the flowchart of our proposed algorithm is shown in Fig. 10, where the DT-based x mode decision model is shows in Fig. 4. It should be noted that different decision models are applied for different mode separately, and the average prediction accuracy will be discussed in Section IV.C as the hit rate for each sequence. In total, 16 DTs are trained by selecting different features and splitting criterions, and they contain 204 values for splitting criterions. By storing each value using double-precision floating point which requires 8 bytes (B) in C language, those values only take up 1.59 KB. A CU goes through a mode decision model before



CU start
Go to next depth level
IBCPredictor
DT-based Intra mode decision model
Label=ω_{Intra}?
No
Yes
Intra mode
DT-based IBCM&S decision model
Label=$\omega_{IBCM\&S}$?
No
Yes
IBCM&S
DT-based IBCSearch decision model
Label=$\omega_{IBCSearch}$?
No
Yes
IBCSearch
DT-based PLT mode decision model
Label=ω_{PLT}?
No
Yes
PLT mode
d_{SCC} = 3?
Yes
No
Early CU partitioning termination in (6)?
No
Yes
CU end
SCC encoding process
Additional procedure of the proposed algorithm

FIGURE 10. Flowchart of the proposed FHST, and DT-based *x* mode decision model is illustrated in Fig. 4.

checking a mode. If the label given by the decision model is ω_x, the mode x would be checked. Otherwise, it would be skipped. Besides, when the CU partitioning termination rule described in (6) is satisfied, the encoding process of this CU is finished.

IV. EXPERIMETNAL RESULTS AND DISCUSSIONS

To evaluate the performance of the proposed FHST, extensive experiments were conducted on a HP EliteDesk 800 G1 computer with a 64-bit Microsoft Windows 10 OS running on an Intel Core i7-4790 CPU of 3.6 GHz and 32.0 GB RAM. Since there are many legacy screen content videos encoded by HEVC in YUV4:2:0 format, we conducted various experiments to evaluate the transcoding performance of YUV4:2:0 screen content videos. We implemented our proposed FHST in HM-16.12 and HM-16.12+SCM-8.3, and all experiments were conducted with QPs of 22, 27, 32 and 37 under AI configuration and common test conditions (CTC) [28]. All test sequences were firstly encoded by HM-16.12 with QPs of 22, 27, 32 and 37, and then the decoded frames were re-encoded by the proposed FHST with the same QPs. The coding efficiency and re-encoding time of the proposed FHST were compared with the conventional brute-force transcoder (CBFT), and they are measured by BDBR and re-encoding time increase, ΔTime, in percentage (%). It is noted that BDBR is calculated by comparing the HEVC decoded video and the final transcoded video.

A. CONFIDENCE THRESHOLD DETERMINATION

To achieve a good trade-off between the coding efficiency and computational complexity, two confidence thresholds α and β were set in the Spatial-Info classifier and x mode classifier, respectively. If the accuracy of a decision made by the Spatial-Info classifier or x mode classifier is smaller than the value of α or β, the class label for mode x is set to ω_x regardless of its outcome. In this sub-section, the performance of our proposed FHST is investigated by adopting a greedy searching strategy. The default values of α and β are always 0.5 in a DT. First, the value of α was fine-tuned with $\beta = 0.5$. Second, the value of β was fine-tuned with α set to the value providing the best performance. The performances with different values of α and β are shown in Table 2. It is observed that FHST with the default values of confidence thresholds ($\alpha = 0.5$, $\beta = 0.5$) provides 53.19% re-encoding time reduction with BDBR increased by 1.89%. By adjusting the values of α and β ($\alpha = 0.75$, $\beta = 0.65$), BDBR increment is reduced to 1.03%, and the re-encoding time is reduced by 46.46%.

Besides, it is also observed from the Table 2 that the proposed FHST provides similar performance for sequences used for training (T) and sequences not used for training (NT). For example, with $\alpha = 0.75$ and $\beta = 0.5$, the average re-encoding time reduction of T and NT sequences are 53.06% and 50.10%, while the BDBR of T and NT sequences are increased by 1.45% and 1.24%, respectively. It proves that the training process of our proposed FHST does not run into

TABLE 2. Performance of the proposed FHST for YUV 4:2:0 sequences with different values of confidence thresholds.

Sequences	Tuning of α (β=0.5)						Tuning of β (α=0.75)					
	α=0.50		α=0.75		α=0.95		β=0.55		β=0.60		β=0.65	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Console (*T*)	3.39	-40.86	2.89	-39.57	2.51	-33.33	2.83	-39.26	2.82	-39.01	2.57	-33.54
Desktop (*T*)	1.18	-40.67	1.15	-40.38	0.99	-32.14	1.12	-40.27	1.12	-39.21	0.55	-33.54
Map (*T*)	1.29	-59.36	0.11	-57.02	0.17	-38.11	0.16	-54.92	0.16	-54.81	0.03	-54.58
MissionControlClip2 (*T*)	1.96	-55.88	1.11	-53.46	0.88	-43.40	1.15	-52.72	0.99	-52.17	0.89	-49.10
Robot (*T*)	1.94	-78.58	2.00	-74.87	2.01	-53.17	2.04	-72.31	2.05	-72.22	2.09	-73.85
BasketballScreen (*NT*)	2.06	-55.17	1.15	-52.95	0.77	-39.80	1.12	-51.48	1.02	-51.07	0.80	-48.44
ChineseEditing (*NT*)	1.11	-37.68	0.80	-36.16	0.57	-29.01	0.77	-35.68	0.75	-35.05	0.45	-29.50
ChinaSpeed (*NT*)	1.36	-63.28	0.99	-60.02	0.86	-46.05	0.97	-58.55	0.97	-58.54	0.93	-57.19
FlyingGraphics (*NT*)	2.74	-41.28	2.08	-40.13	1.28	-31.24	1.95	-39.18	1.81	-38.52	1.49	-32.24
MissionControlClip3 (*NT*)	2.33	-50.53	1.70	-48.81	1.42	-37.29	1.69	-47.97	1.60	-46.97	1.31	-43.07
Programming (*NT*)	1.70	-47.30	0.88	-45.88	0.60	-37.50	0.87	-45.04	0.81	-44.62	0.67	-40.39
SlideShow (*NT*)	1.50	-71.22	0.71	-68.71	0.70	-59.47	0.66	-67.00	0.60	-66.92	0.63	-66.16
WebBrowsing (*NT*)	1.92	-49.59	1.62	-48.15	1.42	-39.32	1.87	-47.49	1.74	-47.02	0.98	-42.32
Average (*T*)	1.95	-55.07	1.45	-53.06	1.31	-40.03	1.46	-51.90	1.43	-51.48	1.23	-48.92
Average (*NT*)	1.84	-52.00	1.24	-50.10	0.95	-39.96	1.24	-49.05	1.16	-48.59	0.91	-44.91
Average (*ALL*)	1.89	-53.19	1.32	-51.24	1.09	-39.99	1.32	-50.14	1.26	-49.70	1.03	-46.46

T: Sequence used for training. *NT*: Sequence not used for training.

TABLE 3. Performance comparison of the proposed FHST with different fast SCC encoding algorithms for YUV 4:2:0 sequences.

Sequences	CBFT + Zhang [24]		CBFT + Lei [25]		CBFT + Yang [26]		Proposed FHST	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Console (*T*)	2.61	-24.29	2.25	-18.97	5.43	-38.62	2.89	-39.57
Desktop (*T*)	1.60	-27.90	1.32	-19.59	4.10	-41.38	1.15	-40.38
Map (*T*)	1.58	-27.64	4.05	-21.45	5.13	-32.55	0.11	-57.02
MissionControlClip2 (*T*)	2.19	-28.03	2.71	-28.18	2.69	-40.41	1.11	-53.46
Robot (*T*)	11.37	-15.22	12.85	-34.15	4.90	-36.15	2.00	-74.87
BasketballScreen (*NT*)	1.69	-29.43	1.41	-24.06	3.17	-37.80	1.15	-52.95
ChineseEditing (*NT*)	0.52	-31.19	0.59	-17.52	3.32	-35.54	0.80	-36.16
ChinaSpeed (*NT*)	0.82	-22.00	0.47	-27.31	1.32	-41.14	0.99	-60.02
FlyingGraphics (*NT*)	0.41	-12.75	1.10	-16.97	3.68	-36.31	2.08	-40.13
MissionControlClip3 (*NT*)	1.57	-26.09	1.78	-22.26	2.95	-37.19	1.70	-48.81
Programming (*NT*)	1.29	-27.91	1.57	-23.32	4.00	-38.57	0.88	-45.88
SlideShow (*NT*)	1.89	-43.19	4.88	-51.60	4.05	-57.34	0.71	-68.71
WebBrowsing (*NT*)	2.35	-37.28	2.85	-24.01	5.58	-40.34	1.62	-48.15
Average (*T*)	3.87	-24.62	4.64	-24.47	4.45	-37.82	1.45	-50.06
Average (*NT*)	1.31	-28.73	1.83	-25.88	3.50	-40.53	1.24	-52.10
Average (*ALL*)	2.30	-27.15	2.91	-25.34	3.87	-39.49	1.32	-51.24

overfitting, and it can be well applied to other screen content sequences. In the following sub-sections, we set α to 0.75 and β to 0.5 for further discussions, which provides 51.24% encoding time reduction with 1.32% increase in BDBR on average.

B. PERFORMANCE COMPARISON OF YUV4:2:0 FORMAT

To evaluate the efficiency of FHST, we compared it with CBFT in which the original SCC encoder of CBFT in Fig. 1 is replaced by the fast SCC encoding algorithms in [24]–[26]. It is noted that the work in [27], which is the only existing fast HEVC to SCC transcoding algorithm, is only worked for sequences in YUV 4:4:4 format, and the comparison will be shown later in Section IV.F. Table 3 shows the performance comparisons based on HM-16.12 and HM-16.12+SCM-8.3. Compared with the fast SCC encoding algorithms in [24]–[26], the proposed FHST additionally utilizes features from the HEVC decoder to improve prediction accuracy, and it is observed that performance of our proposed FHST outperforms the fast SCC encoding algorithms [24]–[26]. On average, the proposed FHST provides 51.24% re-encoding time reduction with a negligible increase in BDBR of 1.32%. Comparatively, Zhang *et al.*'s algorithm [24], Lei *et al.*'s algorithm [25] and Yang *et al.*'s algorithm [26] all bring very high increase in BDBR, where 27.15%, 25.34% and 39.49% re-encoding time is reduced with 2.30%, 2.91% and 3.87% increase in BDBR, respectively. The fast SCC encoding algorithms [24]–[26] only utilize features from the SCC encoder, and they heavily rely on the assumption that the computer-generated content is noiseless. However, this assumption does not hold for decoded videos due to the lossy encoding and decoding of HEVC. Therefore, the fast SCC encoding algorithms [24]–[26] provide less improvement.

TABLE 4. Performance of the proposed algorithm with other structures.

Sequences	FHST2		FHST3	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Console (*T*)	8.90	-50.22	3.00	-25.73
Desktop (*T*)	4.95	-46.04	0.50	-24.01
Map (*T*)	5.14	-71.20	1.38	-47.84
MissionControlClip2 (*T*)	6.62	-61.22	1.68	-44.98
Robot (*T*)	3.95	-78.73	2.30	-67.94
BasketballScreen (*NT*)	8.12	-58.85	1.71	-41.86
ChineseEditing (*NT*)	3.71	-43.45	0.64	-21.85
ChinaSpeed (*NT*)	6.00	-68.85	0.83	-52.27
FlyingGraphics (*NT*)	8.97	-48.04	2.18	-22.73
MissionControlClip3 (*NT*)	7.39	-53.60	1.99	-35.34
Programming (*NT*)	4.68	-55.15	1.49	-32.57
SlideShow (*NT*)	6.61	-74.33	1.40	-63.47
WebBrowsing (*NT*)	2.82	-50.03	1.18	-32.69
Average	5.99	-58.43	1.56	-39.48

TABLE 5. Performance of each proposed technique for YUV 4:2:0 sequences.

Sequences	Early CU partitioning termination		Flexible mode decision	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Console (*T*)	1.82	-9.06	2.32	-37.20
Desktop (*T*)	0.25	-5.95	1.09	-39.03
Map (*T*)	0.35	-14.03	-0.09	-52.90
MissionControlClip2 (*T*)	0.37	-23.11	1.65	-44.50
Robot (*T*)	2.55	-34.82	1.20	-62.00
BasketballScreen (*NT*)	0.15	-16.11	1.62	-48.28
ChineseEditing (*NT*)	0.13	-6.21	0.87	-34.52
ChinaSpeed (*NT*)	0.23	-24.12	0.94	-50.99
FlyingGraphics (*NT*)	0.31	-5.65	2.08	-38.93
MissionControlClip3 (*NT*)	0.83	-13.26	1.65	-44.29
Programming (*NT*)	0.18	-14.25	0.89	-41.27
SlideShow (*NT*)	0.30	-46.65	0.73	-47.42
WebBrowsing (*NT*)	0.35	-11.79	1.74	-45.13
Average	0.60	-17.31	1.28	-45.11

C. DISCUSSION ON THE STRUCTURE OF x MODE DECISION MODEL

Apart from the x mode decision model in Fig. 4, there are other possible structures, such as training a single DT by utilizing all features (FHST2) and applying the Spatial-Info classifier to all depth levels (FHST3). For FHST2, all features in Section III.B.2 are used to train the x mode classifier without the Spatial-Info classifier. For FHST3, the condition for checking the Spatial-Info classifier in Fig. 4 is removed. However, it is found that they fail to achieve a good tread-off between the re-encoding time and RD performance. The performances of FHST2 and FHST3 are shown in Table 4 with the default values of confidence thresholds ($\alpha = 0.5$, $\beta = 0.5$). It is observed that FHST2 brings a very high increase in BDBR of 5.99%. The reason is that FHST2 is strongly affected by error propagation due to the adoption of spatial features into the single DT. For example, the DT would directly skip IBC and PLT modes for the current CUs if the neighbor CUs are NIBs. Comparatively, FHST avoids the error propagation by treating the spatial features as additional features, and they are utilized to train another DT. On the other hand, FHST3 needs to check more mode candidates by applying the Spatial-Info classifiers to all depth levels. Therefore, it provides a limited encoding time reduction of 39.48%. Comparatively, the proposed FHST achieves a good trade-off between the encoding time and RD performance by applying the Spatial-Info classifier to the last depth level. With the default values of confidence thresholds, FHST provides 53.19% re-encoding time reduction with BDBR increased by 1.89%. By setting α to 0.75 and β to 0.5, FHST has even smaller increase of BDBR than FHST3, where 51.24% re-encoding time is reduced with BDBR increased by 1.32%. Therefore, we adopt it as the optimal structure to the x mode decision model.

D. PERFORMANCE OF THE INDIVIDUAL TECHNIQUE

In this sub-section, the performances of the early CU partitioning termination technique and the flexible mode decision technique are evaluated separately, and the results are shown in Table 5. It is observed that the proposed flexible mode decision technique achieves 45.11% re-encoding time reduction with BDBR increased by 1.28% on average. Besides, the early CU partitioning termination technique provides 17.31% re-encoding time reduction while BDBR is increased by 0.60% on average. More specifically, it provides the largest re-encoding time reduction of 46.65% for "SlideShow". The reason is that "SlideShow" contains many smooth areas, which are encoded with many large CUs by HEVC. Therefore, with the help of the decoder side information of HEVC, many CU partitions in SCC are early terminated, and it leads to large re-encoding time reduction. Furthermore, to understand the re-encoding time reduction of our proposed FHST in low and high bit rate cases, we compared the re-encoding time reduction with different QPs. Fig. 11 shows the results of 4 sequences including "BasketballScreen", "ChinaSpeed", "Desktop" and "WebBrowsing", and similar results are observed for other sequences. The re-encoding time reduction provided by the early CU partitioning termination technique, flexible mode decision technique, and overall algorithm are shown in Fig. 11(a), (b) and (c), respectively. It is observed in Fig. 11(a) that the early CU partitioning termination technique provides more re-encoding time reduction as QP increases. The reason is that many CUs are encoded with large sizes by HEVC with a large value of QP, and more CUs are early terminated in SCC by using the early CU partitioning termination technique. On the contrary, the re-encoding time reduction provided by the flexible mode decision technique decreases as QP increases. The reason is that static features contain more noise as QP increases, which leads to the decrease of the decision accuracy. Although the re-encoding reduction provided by each sub-algorithm is different as QP changes, it is observed in Fig. 11(c) that the re-encoding reduction of the overall algorithm varies little across different values of QP. Therefore, our proposed FHST has stable performance as bit rate varies.

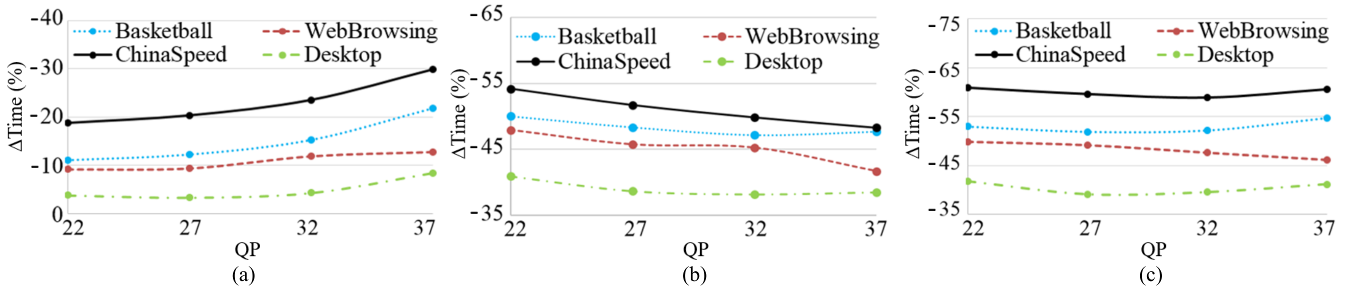


FIGURE 11. Re-encoding time reduction of (a) early CU partitioning termination technique, (b) flexible mode decision technique, and (c) the proposed overall algorithm over 4 QPs.

TABLE 6. Hit rates of the proposed techniques for YUV 4:2:0 sequences.

Sequences	Early CU partitioning termination (%)				Flexible mode decision (%)			
	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37
Console (*T*)	95.78	93.67	92.72	96.30	91.27	90.43	87.97	88.81
Desktop (*T*)	98.03	98.25	98.40	97.78	92.90	91.71	88.37	85.96
Map (*T*)	98.65	98.62	98.10	98.02	94.40	95.00	94.79	94.45
MissionControlClip2 (*T*)	97.63	97.69	97.96	98.18	92.11	90.88	92.91	93.85
Robot (*T*)	97.46	97.32	97.64	97.84	95.89	94.74	94.64	93.71
BasketballScreen (*NT*)	97.49	98.06	98.08	98.00	92.90	91.54	90.38	90.51
ChineseEditing (*NT*)	98.09	98.34	98.50	98.54	91.12	90.81	90.91	90.50
ChinaSpeed (*NT*)	99.06	94.44	94.75	94.71	94.71	92.90	91.30	89.44
FlyingGraphics (*NT*)	98.50	98.52	98.28	97.20	92.16	90.77	89.03	88.87
MissionControlClip3 (*NT*)	98.05	98.14	98.27	98.19	91.92	91.97	91.73	91.39
Programming (*NT*)	97.50	97.83	98.15	97.84	93.58	92.84	91.79	91.12
SlideShow (*NT*)	98.90	98.91	99.06	99.04	95.55	95.81	95.98	96.23
WebBrowsing (*NT*)	98.03	97.78	98.50	98.46	92.81	92.07	88.12	87.81
Average (*T*)	97.51	97.11	96.96	97.62	93.31	92.55	91.74	91.36
Average (*NT*)	98.20	97.75	97.95	97.75	93.09	92.33	91.16	90.74
Average (*ALL*)	97.94	97.51	97.57	97.70	93.17	92.42	91.37	90.97

Another way to evaluate the proposed FHST is to investigate the hit rates of the proposed techniques compared with the CBFT transcoder. In this sub-section, the hit rates of the early CU partitioning termination and flexible mode decision techniques are given by calculating the percentages of the areas encoded by the same mode as in CBFT, and the results are shown in Table 6. It is observed that the average hit rates of the proposed early CU partitioning termination and flexible mode decision techniques are all above 90%. More specifically, the hit rate of the early CU partitioning termination technique varies from 92.72% to 99.06%, while the hit rate of the flexible mode decision technique varies from 85.96% to 96.23% for different sequences with different QPs. Besides, the average hit rate of the early CU partitioning termination technique varies little underdifferent QPs, while the average hit rate of the flexible mode decision technique is increased from 90.97% to 93.17% as QP gets smaller. It is due to the fact that the decoded HEVC videos contain less noise as QP gets smaller, and the static features utilized in our proposed mode decision models can describe the CU content characteristics more precisely.

Since the flexible mode decision technique contributes significantly to our proposed FHST, the mode decision of FHST is further studied. The decision of each mode is visualized, and it is compared with the mode decision made by CBFT. Fig. 12 and Fig. 13 show the mode decision of a region in "ChinaSpeed" and a region in "Programming", respectively, at the depth level of 2 and QP of 22. Fig. 12(a) and Fig. 13(a) show the optimal mode decided by CBFT, where Intra, IBC and PLT modes are denoted by blue, purple and yellow blocks, respectively. It should be noted that for CUs without any denoted color, they are not encoded at the depth level of 2. Fig. 12(b)-(d) and Fig. 13(b)-(d) show the Intra mode skipped CUs, IBC mode skipped CUs, PLT mode skipped CUs decided by our proposed flexible mode decision technique, where CUs with incorrectly and correctly skipped modes are denoted by red shaded blocks and green shaded blocks, respectively. It is observed in Fig. 12(b) that Intra mode is skipped for many NIBs, because d_{HEVC}^{avg} of these CUs are not equal to the current depth level in SCC, as analyzed in Section III.B.2.a. Besides, IBC mode and PLT mode are skipped for many CUs as shown in Fig.12(c) and (d). However, almost all SCBs are well detected and decided to check PLT mode by the proposed FHST. It is also noted that some smooth CUs need to check all modes, as shown in Fig.12(b)-(d). The reason is that those smooth CUs may select any mode in the training frames, so that it is difficult to make flexible mode decisions. When compared with the mode decisions made by CBFT, only 4 CUs are incorrectly skipped, while the remaining 106 CUs

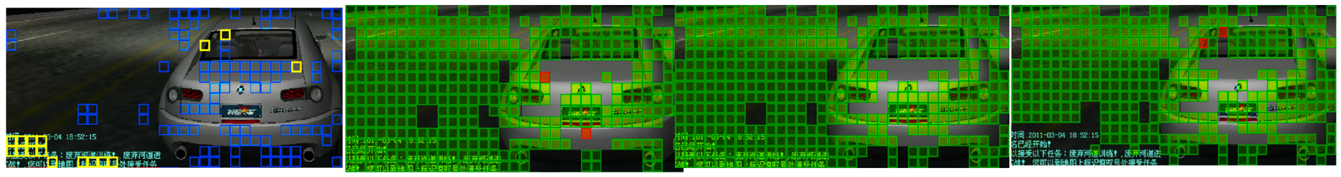


FIGURE 12. Mode decisions of a region in "ChinaSpeed" with the depth level of 2 and QP of 22. (a) Mode decision of CBFT, where Intra, IBC and PLT modes are denoted by blue, purple and yellow blocks. (b) Intra mode skipped CUs, (c) IBC mode skipped CUs, (d) PLT mode skipped CUs decided by our proposed FHST, where CUs with incorrectly and correctly skipped mode are denoted by red shaded blocks and green shaded blocks, respectively.

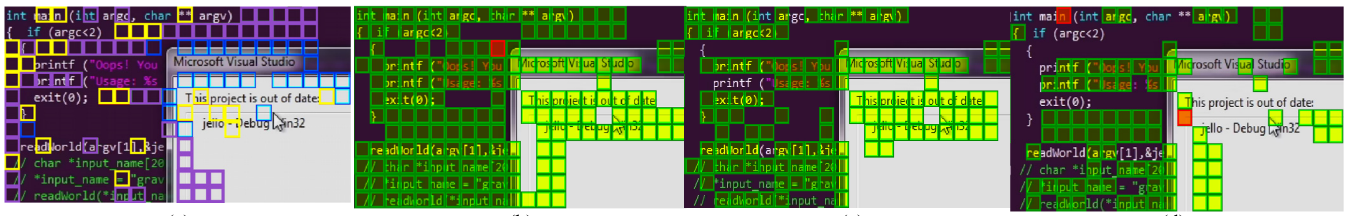


FIGURE 13. Mode decisions of a region in "Programming" with the depth level of 2 and QP of 22. (a) Mode decision of CBFT, where Intra, IBC and PLT modes are denoted by blue, purple and yellow blocks. (b) Intra mode skipped CUs, (c) IBC mode skipped CUs, (d) PLT mode skipped CUs decided by our proposed FHST, where CUs with incorrectly and correctly skipped mode are denoted by red shaded blocks and green shaded blocks, respectively.

are correctly skipped and then encoded by the optimal modes correctly.

For the region containing many SCBs in "Programming", it is observed in Fig. 13(b) that almost all SCBs are well detected and Intra mode are skipped for them. Besides, it is observed in Fig. 13(c) and (d) that many SCBs are decided to check one mode either from IBC or PLT mode, so that the re-encoding time is further reduced when compared with the fast mode decision algorithms [25]–[27] that only perform CU type classification. When compared with the mode decisions made by CBFT in Fig 13(a), only 3 CUs are incorrectly skipped, while the remaining 91 CUs are encoded by their optimal modes correctly. Therefore, many redundant mode candidates are skipped by the proposed FHST, while the optimal modes are well kept.

E. DISCUSSION ON FEATURE IMPORTANCE

Our proposed FHST utilizes features from 4 categories to reduce the re-encoding time of SCC, which are features from the HEVC decoder, static features, dynamic features and spatial features. In this sub-section, we discuss the importance of each feature category by removing it from our proposed FHST firstly and then observing the performance improvement when it is added back. In general, better performance is denoted by larger re-encoding time reduction and smaller BDBR increase. Therefore, we adopt a similar performance factor, *PFactor*, as in [34] to denote the coding performance

$$PFactor = -\frac{\Delta\text{Time}}{\text{BDBR}}. \qquad (19)$$

A larger value of *PFactor* represents better performance. Then based on *PFactor*, we calculate the importance factor, *IFactor*, of each feature category by

$$IFactor = \frac{PFactor_{FA} - PFactor_{FR}}{PFactor_{FR}} \qquad (20)$$

where $PFactor_{FR}$ and $PFactor_{FA}$ are the performance factors of feature removed and feature added back transcoders, respectively. Table 7 presents the results of the proposed FHST when either one category of features is removed. It should be noted that removing spatial features means disabling the Spatial-Info classifier in Fig. 4. It is observed that all feature categories are helpful for improving coding performance, and the transcoder with all features implemented has the best performance with *PFactor* of 38.82. For the transcoder without spatial features, decoder features, dynamic features and static features, 56.80%, 44.45%, 49.06% and 56.71% re-encoding time are saved while the BDBR is increased by 4.48%, 2.94%, 1.71% and 3.19%, respectively. Therefore, the most important feature category to the proposed FHST is spatial features, and then followed by decoder features, static features and dynamic features, whose *IFactors* are 2.06, 1.57, 1.18 and 0.83, respectively.

F. PERFORMANCE COMPARISON OF YUV4:4:4 FORMAT

In this sub-section, the proposed FHST is further extended to support fast transcoding of screen content videos in YUV4:4:4 format, where "Console", "Desktop", "Map", "MissionControlClip2", "Robot" were used to generate training data, and α is set to 0.75, β is set to 0.5. Similarly, the fast algorithms in [24]–[26] are used to replace the original SCC encoder of CBFT in Fig. 1 for comparison. Besides, we also compared FHST with the fast HEVC to SCC transcoding algorithm [27], which is only designed for YUV4:4:4 format. Considering that Duanmu *et al.*'s

TABLE 7. Performance of the proposed transcoder for YUV 4:2:0 sequences with different feature combination.

Sequences	No spatial features		No decoder features		No dynamic features		No static features		All features	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Console (*T*)	7.64	-47.86	2.18	-36.30	3.57	-38.49	6.22	-47.88	2.89	-39.57
Desktop (*T*)	3.57	-45.50	1.28	-38.05	1.50	-36.56	3.34	-49.20	1.15	-40.38
Map (*T*)	5.01	-62.03	2.58	-56.83	1.62	-54.42	3.14	-62.92	0.11	-57.02
MissionControlClip2 (*T*)	4.50	-58.86	2.95	-44.05	1.50	-51.18	2.82	-57.09	1.11	-53.46
Robot (*T*)	2.24	-77.97	6.75	-58.27	1.97	-73.92	1.86	-76.56	2.00	-74.87
BasketballScreen (*NT*)	4.63	-58.30	3.50	-47.58	1.63	-49.84	3.22	-57.13	1.15	-52.95
ChineseEditing (*NT*)	3.41	-43.47	1.84	-34.05	0.96	-35.57	2.47	-44.26	0.80	-36.16
ChinaSpeed (*NT*)	2.50	-66.12	1.44	-52.32	1.30	-58.16	1.41	-64.07	0.99	-60.02
FlyingGraphics (*NT*)	7.06	-49.20	2.67	-39.00	2.28	-38.67	5.02	-47.29	2.08	-40.13
MissionControlClip3 (*NT*)	4.43	-52.92	2.68	-44.46	1.73	-45.68	3.19	-54.04	1.70	-48.81
Programming (*NT*)	3.89	-51.72	2.21	-40.24	1.24	-42.97	2.30	-52.06	0.88	-45.88
SlideShow (*NT*)	4.97	-72.30	5.99	-43.01	1.17	-66.99	2.30	-71.28	0.71	-68.71
WebBrowsing (*NT*)	4.38	-52.11	2.10	-43.66	1.77	-45.29	4.24	-53.43	1.62	-48.15
Average	4.48	-56.80	2.94	-44.45	1.71	-49.06	3.19	-56.71	1.32	-51.24
PFactor	12.68		15.12		28.69		17.78		38.82	
IFactor	Spatial features 2.06		Decoder features 1.57		Dynamic features 0.83		Static features 1.18			

TABLE 8. Performance comparison of the proposed transcoder with different fast SCC encoding algorithms for YUV 4:4:4 sequences.

Sequences	CBFT + Zhang [24]		CBFT + Lei [25]		CBFT + Yang [26]		Proposed transcoder	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Console (*T*)	1.67	-20.26	2.24	-21.93	4.79	-30.15	2.36	-50.24
Desktop (*T*)	1.36	-30.16	1.35	-23.21	4.08	-35.06	0.52	-56.92
Map (*T*)	1.21	-28.28	3.51	-20.04	7.26	-15.73	1.00	-45.79
MissionControlClip2 (*T*)	2.22	-29.55	2.84	-28.02	3.87	-29.56	1.50	-54.29
Robot (*T*)	10.89	-12.63	12.67	-30.92	5.47	-19.31	1.71	-60.26
BasketballScreen (*NT*)	1.34	-30.95	1.19	-22.38	3.80	-25.11	0.92	-52.01
ChineseEditing (*NT*)	0.33	-22.84	0.61	-19.42	3.63	-24.90	0.59	-47.26
EBURainFruits (*NT*)	7.38	-18.61	7.34	-34.72	2.88	-20.61	2.23	-67.92
FlyingGraphics (*NT*)	0.45	-4.39	1.08	-19.14	4.56	-26.84	1.03	-45.53
Kimono1(*NT*)	10.29	-3.79	8.39	-41.82	4.18	-27.14	1.39	-56.99
MissionControlClip3 (*NT*)	1.83	-26.93	1.43	-23.64	3.08	-29.30	1.24	-54.57
Programming (*NT*)	1.28	-28.42	1.76	-22.19	7.27	-26.35	1.08	-50.74
SlideShow (*NT*)	1.99	-41.03	1.76	-48.79	5.81	-45.94	1.37	-63.83
WebBrowsing (*NT*)	1.57	-40.16	3.21	-25.38	5.65	-34.30	0.50	-58.80
Average (*T*)	3.47	-24.18	4.52	-24.82	5.09	-25.96	1.42	-53.50
Average (*NT*)	2.94	-24.12	2.97	-28.61	4.54	-28.94	1.15	-55.29
Average (*ALL*)	3.13	-24.14	3.53	-27.26	4.74	-27.88	1.25	-54.65

algorithm [27] is the only existing fast HEVC to SCC transcoding algorithm, and it was implemented in HM-16.4 [35] and HM-16.4 + SCM-4.0 [36], we re-implemented all other algorithms in the same reference software as Duanmu *et al.*'s algorithm [27] to make fair comparisons.

Table 8 shows the comparisons of the proposed FHST with the fast SCC encoding algorithms [24]–[26] under CTC [28]. It is also observed that the performance of FHST is much better than the fast SCC encoding algorithms [24]–[26]. For the T sequences, 53.50% re-encoding time is saved with 1.42% increase in BDBR. For the NT sequences, similar performance is obtained, where 55.29% re-encoding time is reduced with BDBR increased by 1.15%. It again proves that the proposed FHST is generalizable to the sequences which are not used in training. On average, the proposed FHST provides 54.65% re-encoding time reduction with a negligible increase in BDBR of 1.25%. Comparatively, Zhang *et al.*'s algorithm [24], Lei *et al.*'s algorithm [25] and Yang *et al.*'s algorithm [26] provide 24.14%, 27.26% and 27.88% re-encoding time reduction with BDBR increased by 3.13%, 3.53% and 4.74%, respectively. Then, based on the same reference software of HEVC and SCC, we made an indirect comparison between our proposed FHST and the only existing fast HEVC to SCC transcoding algorithm [27], and the results are presented in Table 9. It is observed when compared with CBFT, Duanmu *et al.*'s algorithm [27] achieves 47.93% re-encoding time reduction for their selected sequences while BDBR is increased by 2.14%. Comparatively, our proposed FHST achieves 54.01% re-encoding time reduction for their selected sequences while BDBR is only increased by 1.11%. Compared with Duanmu *et al.*'s algorithm [27] which only utilizes features from the HEVC decoder and static features, our proposed FHST additionally utilizes spatial features and dynamic features, so that more accurate decision is provided. Besides, Duanmu *et al.*'s algorithm [27] always checks both IBC and PLT modes for SCBs. However, we allow the case that only

TABLE 9. Performance comparison of the proposed transcoder with other transcoder for YUV 4:4:4 sequences.

Sequences	Duanmu [27]		Proposed transcoder	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Console (*T*)	1.85	-49.0	2.36	-50.24
Desktop (*T*)	1.72	-48.1	0.52	-56.92
Map (*T*)			1.00	-45.79
MissionControlClip2 (*T*)			1.50	-54.29
Robot (*T*)			1.71	-60.26
BasketballScreen (*NT*)	3.13	-46.1	0.92	-52.01
ChineseEditing (*NT*)			0.59	-47.26
EBURainFruits (*NT*)			2.23	-67.92
FlyingGraphics (*NT*)	1.94	-50.1	1.03	-45.53
Kimono1(*NT*)			1.39	-56.99
MissionControlClip3 (*NT*)			1.24	-54.57
Programming (*NT*)	1.05	-42.9	1.08	-50.74
SlideShow (*NT*)	2.21	-51.4	1.37	-63.83
WebBrowsing (*NT*)	3.08	-47.9	0.50	-58.80
Average ([27]'s sequences)	2.14	-47.93	1.11	-54.01
Average (*ALL*)			1.25	-54.65

one mode is checked for SCBs, as observed in Fig. 12(c), (d) and Fig. 13(c), (d), so that higher re-encoding time reduction is provided.

V. CONCLUSIONS

In this paper, a fast HEVC to SCC transcoder FHST is proposed by early CU partitioning termination and flexible mode decision. Four categories of features are collected from both the HEVC decoder side and the SCC encoder side to simplify the transcoding process. First, an early CU partitioning termination technique is proposed to map the optimal CU size from HEVC to SCC. Then, a flexible encoding structure is proposed where DTs are generated to check each mode candidate adaptively in SCC. With the help of various features from the four categories and the flexible encoding structure, higher re-encoding time can be reduced with less RD performance loss compared with other algorithms. Experimental results show that the proposed FHST provides 51.24% and 54.65% re-encoding time reduction with a negligible increase in BDBR of 1.32% and 1.25% for YUV 4:2:0 and YUV 4:4:4 screen content sequences, respectively.

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] J. Vanne, M. Viitanen, T. D. Hamalainen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1885–1898, Dec. 2012.

[3] 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.

[4] 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.*, Snowbird, UT, USA, Apr. 2015, pp. 273–282.

[5] X. Xu *et al.*, "Intra block copy in HEVC screen content coding extensions," *IEEE J. Emerg. Sel. Topic Circuits Syst.*, vol. 6, no. 4, pp. 409–419, Dec. 2016.

[6] Z. Ma, W. Wang, M. Xu, and H. Yu, "Advanced screen content coding using color table and index map," *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4399–4412, Oct. 2014.

[7] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Exploiting inter-layer correlations in scalable HEVC for the support of screen content videos," in *Proc. Int. Conf. Digit. Signal Process.*, Hong Kong, Aug. 2016, pp. 1–5.

[8] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document VCEG-M33, VCEG, Austin, TX, USA, Mar. 2001.

[9] J. D. Cock, S. Notebaert, P. Lambert, and R. V. D. Walle, "Architectures for fast transcoding of H.264/AVC to quality-scalable SVC streams," *IEEE Trans. Multimedia*, vol. 11, no. 7, pp. 1209–1224, Nov. 2009.

[10] T.-K. Lee, C.-H. Fu, Y.-L. Chan, and W.-C. Siu, "A new motion vector composition algorithm for fast-forward video playback in H.264," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Paris, France, May/Jun. 2010, pp. 3649–3652.

[11] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "Low-complexity and high-quality frame-skipping transcoder for continuous presence multipoint video conferencing," *IEEE Trans. Multimedia*, vol. 6, no. 1, pp. 31–46, Feb. 2006.

[12] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "New architecture for dynamic frame-skipping transcoder," *IEEE Trans. Image Process.*, vol. 11, no. 8, pp. 886–900, Aug. 2002.

[13] H. Shu and L.-P. Chau, "The realization of arbitrary downsizing video transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 540–546, Apr. 2006.

[14] D. Xu and P. Nasiopoulos, "Logo insertion transcoding for H.264/AVC compressed video," in *Proc. IEEE Int. Conf. Image Process.*, Nov. 2009, pp. 3693–3696.

[15] T. Shanableh, T. May, and F. Ishtiaq, "Error resiliency transcoding and decoding solutions using distributed video coding techniques," *Signal Process. Image Commun.*, vol. 23, no. 8, pp. 610–623, 2008.

[16] Y. L. Chan, H. K. Cheung, and W. C. Siu, "Compressed-domain techniques for error-resilient video transcoding using RPS," *IEEE Trans. Image Process.*, vol. 18, no. 2, pp. 357–370, Feb. 2009.

[17] G. Fernandez-Escribano, H. Kalva, P. Cuenca, L. Orozco-Barbosa, and A. Garrido, "A fast MB mode decision algorithm for MPEG-2 to H.264 P-frame transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, pp. 172–185, Feb. 2008.

[18] G. Fernandez-Escribano, H. Kalva, J. Martinez, P. Cuenca, L. Orozco-Barbosa, and A. Garrido, "An MPEG-2 to H.264 video transcoder in the baseline profile," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 5, pp. 763–768, May 2010.

[19] T. Shanableh, E. Peixoto, and E. Izquierdo, "MPEG-2 to HEVC video transcoding with content-based modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 7, pp. 1191–1196, Jul. 2013.

[20] H. Yuan, C. Guo, J. Liu, X. Wang, and S. Kwong, "Motion-homogeneous-based fast transcoding method from H.264/AVC to HEVC," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1416–1430, Jul. 2017.

[21] F. Zhang, Z. Shi, X. Zhang, and Z. Gao, "Fast H.264/AVC to HEVC transcoding based on residual homogeneity," in *Proc. Int. Conf. Audio, Lang. Image Process.*, Shanghai, China, Jul. 2014, pp. 765–770.

[22] A. Nagaraghatta, Y. Zhao, G. Maxwell, and S. Kannangara, "Fast H.264/AVC to HEVC transcoding using mode merging and mode mapping," in *Proc. Int. Conf. Consum. Electron.*, Berlin, Germany, Sep. 2015, pp. 165–169.

[23] P. Xing, Y. Tian, X. Zhang, Y. Wang, and T. Huang, "A coding unit classification based AVC-to-HEVC transcoding with background modeling for surveillance videos," in *Proc. Vis. Commun. Image Process. (VCIP)*, Kuching, Malaysia, Nov. 2013, pp. 1–6.

[24] 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.*, Shanghai, China, Mar. 2016, pp. 1377–1381.

[25] 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.

[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.*, Beijing, China, Sep. 2017, pp. 2468–2472.

[27] F. Duanmu, Z. Ma, W. Wang, M. Xu, and Y. Wang, "A novel screen content fast transcoding framework based on statistical study and machine learning," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, Sep. 2016, pp. 4205–4209.

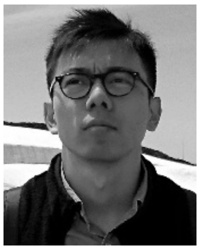
[28] H.-P. Yu, R. Cohen, K. Rapaka, and J.-Z. Xu, *Common Test Conditions for Screen Content Coding*, 24th ed., document JCTVC-X1015-r1, JCT-VC Meeting, Geneva, Switzerland, May 2016.
[29] HM-16.12. *HEVC Test Model Version 16.12*. Accessed: Nov. 27, 2018. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.12/
[30] HM-16.12+SCM-8.3. *HEVC Test Model Version 16.12 Screen Content Model Version 8.3*. Accessed: Nov. 27, 2018. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.12+SCM-8.3/
[31] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[32] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.
[33] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proc. Int. Conf. Artif. Intell.*, 2000, pp. 111–117.
[34] G. Corrêa, 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.
[35] HM-16.4. *HEVC Test Model Version 16.4*. Accessed: Nov. 27, 2018. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.4/
[36] HM-16.4+SCM-4.0. *HEVC Test Model Version 16.4 Screen Content Model Version 4.0*. Accessed: Nov. 27, 2018. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.4+SCM-4.0/

WEI KUANG (S'17) received the B.S. degree from the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University. His research interests include machine learning and deep learning in video coding and video transcoding. He serves as a Reviewer for international journals, including the IEEE Transactions on Circuits and Systems for Video Technology and the *KSII Transactions on Internet and Information Systems*.

YUI-LAM CHAN (S'94–A'97–M'00) received the B.Eng. (Hons.) and Ph.D. degrees from The Hong Kong Polytechnic University, Hong Kong, in 1993 and 1997, respectively.

He joined The Hong Kong Polytechnic University, in 1997, where he is currently an Associate Professor with the Department of Electronic and Information Engineering. He is actively involved in professional activities. He has authored over 110 research papers in various international journals and conferences. His research interests include multimedia technologies, signal processing, image and video compression, video streaming, video transcoding, video conferencing, digital TV/HDTV, 3DTV/3DV, multiview video coding, machine learning for video coding, and future video coding standards, including screen content coding, light-field video coding, and 360° omnidirectional video coding.

Dr. Chan was the Secretary of the 2010 IEEE International Conference on Image Processing. He was the Special Sessions Co-Chair and the Publicity Co-Chair of the 2015 Asia–Pacific Signal and Information Processing Association Annual Summit and Conference. He was also the Technical Program Co-Chair of the 2014 International Conference on Digital Signal Processing. He serves as an Associate Editor for the IEEE Transactions on Image Processing.



SIK-HO TSANG (M'10) received the Ph.D. degree from The Hong Kong Polytechnic University (PolyU), Hong Kong, in 2013.

From 2013 to 2016, he was a Postdoctoral Fellow and was involved in numerous industrial projects for video coding and transcoding. He is currently a Research Fellow with PolyU. He has authored numerous international journals and conference papers. He holds numerous international patents. His current research interests involve data science, machine learning, and deep learning in video coding, such as HEVC, Versatile Video Coding, multiview plus depth coding, screen content coding, light-field video coding, and 360° omnidirectional video coding.

Dr. Tsang serves as a Reviewer for international journals, including the IEEE Transactions on Image Processing, the IEEE Transactions on Circuits and Systems for Video Technology, and the *Journal of Signal Processing: Image Communication* (Elsevier).

WAN-CHI SIU (S'77–M'77–SM'90–F'12–LF'16) received the M.Phil. degree from The Chinese University of Hong Kong, in 1977, and the Ph.D. degree from Imperial College London, in 1984. He was the Chair Professor, the Director of the Signal Processing Research Centre, the Head of the Electronic and Information Engineering Department, and the Dean of the Engineering Faculty, The Hong Kong Polytechnic University, where he is currently an Emeritus Professor. He is an Expert in DSP, fast algorithms, super-resolution imaging, 2D and 3D video coding, and machine learning for object recognition and tracking. He has published 500 research papers (over 200 are international journal papers) and holds nine patents. He is a Fellow of IET and HKIE. He was the President of the Asia–Pacific Signal and Information Processing Association, from 2017 to 2018. From 2000 to 2015, he was also an Independent Non-Executive Director of a publicly-listed video surveillance company and a Convener of the First Engineering/IT Panel of the RAE(1992/1993) in Hong Kong. He is an Outstanding Scholar, with many awards, including the Best Faculty Researcher Award and the IEEE Third Millennium Medal, in 2000. He has organized very successfully over 20 international conferences, including the IEEE society-sponsored flagship conferences, such as the TPC Chair of ISCAS1997, the General Chair of ICASSP2003, and the General Chair of ICIP2010. He was the Vice-President, the Chair of the Conference Board, and the Core Member of the Board of Governors of the IEEE Signal Processing Society, from 2012 to 2014. He is currently a member of the IEEE Fourier Award for the Signal Processing Committee and some other IEEE committees. He is/was a Guest Editor/Subject Editor/AE of the IEEE Transactions on Circuits and Systems, the IEEE Transactions on Image Processing, the IEEE Transactions on Circuits and Systems for Video Technology, and *Electronics Letters*.