Mode Skipping for HEVC Screen Content Coding via Random Forest

Sik-Ho Tsang, *Member, IEEE*, Yui-Lam Chan, *Member, IEEE*, and Wei Kuang, *Student Member, IEEE*

***Abstract*—Screen content coding (SCC) is the extension to High Efficiency Video Coding (HEVC) for compressing screen content videos. New coding tools, intra block copy (IBC) and palette (PLT) modes, are introduced to encode screen content (SC) such as texts and graphics. IBC mode is used for encoding repeating patterns by performing block matching within the same frame, while PLT mode is designed for SC with few distinct colors by coding the major colors and their corresponding locations using an index map. However, the use of IBC and PLT modes increases the encoder complexity remarkably though coding efficiency can be improved. Therefore, we propose to have a mode skipping approach to reduce the encoder complexity of SCC by making use of SC characteristics, neighbor coding unit (CU) correlations, and intermediate cost information via random forest (RF). Detailed feature analyses and sample preparation are also described. A novel hyperparameter tuning approach with the consideration of coding bitrate and encoding time is proposed for RFs at each CU size to further boost the encoding process. Experimental results show that our proposed approach can obtain 45.06% average encoding time reduction with only 1.08% increase in Bjøntegaard delta bitrate (BD-rate). Average encoding time can even be reduced to 58.57% by regulating the hyperparameters.**

***Index Terms*—HEVC, machine learning, random forest, screen content coding, video coding**

I. Introduction

With the recent rapid development of technologies in networking and thin-client devices, computer screen sharing applications have become more popular. They include remote desktop, video conferencing with documents or slides sharing, etc. In addition, there are many television programs (e.g. finance or business news) and many curriculum videos in the Internet that contain mixtures of camera-captured content (CC) and screen content (SC). There will be even more cloud services using screen sharing technology in the near future [1]. These applications result in a substantial demand for the efficient compression of SC. In January 2014, there was Call for Proposal (CfP) [2] of screen content coding (SCC) [3-4] as the extension to High Efficiency Video Coding (HEVC) [5-6] by the Joint Collaborative Team on Video Coding (JCT-VC).

SC is the video content containing computer-generated content such as texts, computer graphics and graphical user interface while CC is the video content captured by camera. Videos sometimes contain a mixture of SC and CC. CC can be encoded by HEVC efficiently. However, SC has discontinuous-tone characteristics [7-8] which is different from CC, such as complex structure with sharp edges, limited number of colors and sometimes high contrast between colors like texts.

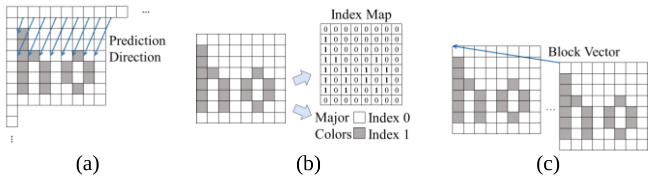

Fig. 1. Illustrations of (a) INTRA, (b) PLT, and (c) IBC modes.

The conventional HEVC intra (INTRA) mode [9-10] shown in Fig. 1(a) uses the neighbor boundary pixels to predict a coding unit (CU) with 33 directional predictions plus planar and DC predictions [9]. To reduce the complexity, rough mode decision (RMD) [10] is performed to select a subset of intra prediction candidates first. Then the optimal one is chosen by rate distortion optimization (RDO) where the full rate-distortion (RD) cost for every candidate in the subset is estimated. However, it cannot efficiently encode the CUs with screen content such as the example shown in Fig. 1(a) since neighbor boundary pixels cannot predict the pixels with abrupt change within the CU. Hence, two new coding tools are introduced in SCC to solve this problem. They are palette (PLT) mode [11] and intra block copy (IBC) mode [12].

For PLT [11] as demonstrated in Fig. 1(b), a CU is separated into color data and structural data. The color data consists of few major colors which are predicted from color table or from neighbor CUs. When the colors are not in the major color tables, they are regarded as escape colors and are explicitly coded. The structural data is then represented by an index map and the corresponding indices are entropy coded. Thus, PLT helps to encode SC which contains few colors like texts and icons. Further improvements were also suggested in [13-16] to further increase the coding efficiency of PLT. IBC mode, as in Fig. 1(c), is a block matching technique to find the repeating patterns within the same frame which frequently occurs in SCs [12]. For instance, repeating numbers and characters can be found within a document and spreadsheet. If IBC is used by one particular

Manuscript received Jun. 2018. The work was supported by the Hong Kong Research Grants Council under Research Grant PolyU 152112/17E.

S.-H. Tsang, Y.-L. Chan, and W. Kuang, are with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: sik-ho.tsang@polyu.edu.hk, enylchan@polyu.edu.hk, wei.kuang@connect.polyu.hk).

CU, each prediction unit (PU) within the CU is encoded with a block vector (BV), as well as the residual signal of that CU which is similar to an inter mode in inter-frame motion estimation. Besides, merge mode and skip mode are also performed in IBC. Further enhancements can be found in [17-21] to improve the coding efficiency of IBC. Moreover, the algorithms in [22] and [23] proposed Pseudo 2D String Matching (P2SM) and Universal String Matching (USM), respectively, to have a more flexible string matching compared with the conventional IBC to further boost the coding efficiency of SCC.

Although PLT and IBC can essentially increase the coding efficiency of SCC, the encoder complexity is impractically high. Therefore, numerous fast approaches [24-40] were proposed to reduce the complexity of SCC. Budagavi *et al.* [24-25] proposed to limit the search area for IBC, skip IBC based on CU size, and use different searching strategies in IBC based on the horizontal and vertical CU activity. The proposed approaches have already been adopted in the current SCC reference software. Full-frame hash search was suggested in [26] by only performing block matching between those block candidates which have the same hash value as the current CU. The hash value is estimated based on the number of color transitions along the row and column. Zhang *et al.* [27] proposed to exploit the CU mode from the collocated CUs of the previous frame. This method is mainly targeted for stationary CUs. Fast CU partitioning based on the CU entropy and the coding bits was designed in [28]. Lei *et al.* [29] proposed to utilize the content property analysis, bits per pixel information as well as neighbor and collocated CUs' depth information to classify CUs into screen content CU (SCCU) or camera-captured content CU (CCCU), and then simplify INTRA mode, mode elimination and fast CU partitioning with numerous pre-defined thresholds. By checking whenever neighbor boundary pixels are exactly the same, we proposed a simple intra prediction (SIP) [30] to skip the RMD [10] and RDO for reducing the complexity of INTRA. IBC and PLT are also skipped when the residual error obtained by INTRA is zero. In [31], we suggested to skip IBC and PLT when every row or column of pixels are equal or it is smooth within the CU by checking if the CU has zero CU activity or low gradient smooth area. Our previous work in [32] suggested using the hash checking during block matching to reduce the complexity within IBC. The algorithm in [33] made use of sharp and directional edges for mode and CU depth skipping. But the results are worse than [30]. For machine learning approaches, we proposed [34] to use Bayesian decision rules based on corner point detection for fast mode decision plus online learning approaches with scene change detection. In [35], a neural network based fast algorithm was proposed to make fast CU partitioning by utilizing features that describe CU statistics and sub-CU homogeneity. However, high RD performance loss is induced by this approach. In [36], a decision tree based classifier was firstly designed to classify CUs into CCCUs and SCCUs, intuitive logics are then applied by only checking INTRA for CCCUs and evaluating IBC and PLT for SCCUs. Besides, to speed up the encoding process of CCCUs, two classifiers were designed to predict the Intra mode direction from 35 prediction modes and early terminate the partitions of CCCUs, respectively. There are still pre-defined thresholds for these classifiers. Yang *et al.* [37] proposed to have two decision tree classifiers for fast CU partitioning and CU type classification. If a CU is classified as a partition CU, it directly goes to the next CU size. Otherwise, it is further classified as a CCCU or SCCU. If it is a SCCU, both IBC and PLT modes are checked. If it is a CCCU, only INTRA mode is tested. Recently, we also proposed two rule-based approaches [38-39], in which the results are not good enough, and one decision tree approach [40] in which handcrafted decisions using neighboring CU modes are still needed.

In this paper, we propose a mode skipping approach for INTRA, IBC and PLT, via random forest (RF), based on the SC characteristics, neighbor CU modes, and intermediate cost information. As machine learning approach is used, our design has no pre-defined thresholds for features. Instead of classifying CU into CCCUs and SCCUs [29,36-37], before checking each mode, we rather perform feature extraction and classification using RF to decide whether the current mode should be skipped or not. Therefore, the latest CU information such as the least RD cost as well as the associated distortion and coding rate before the current mode can be obtained, and are input into RF together with other features for mode skipping decision. So, intuitive logics in [29,36-37] are removed. Moreover, among various machine learning algorithms, we choose to apply RF for our proposed approach. This is because multiple trees are used within a RF for voting such that only the decision with majority votes is performed. This property makes RF become not greedy which can preserve the video quality and coding bitrate while speeding up the encoding process to some extent. There are many successful applications of RF for image/video coding and enhancement. For example, Liu *et al.*, and Huang *et al.* [41-42] used RF for super resolution by enhancing the bicubic interpolated image. Fang *et al.* [43] proposed to have single image depth estimation by learning the structure properties based on RF. Du *et al.* [44] used RF for fast CU partitioning in HEVC, but did not use RF in mode decision since there is only INTRA mode in HEVC. The introduction of IBC and PLT modes make the necessity of a completely new fast mode decision method in SCC as they take up significant amount time for mode selection. To the best of our knowledge, we are the first to use RF for fast mode decision in SCC.

The rest of the paper is organized as follows. We start by introducing the SCC intra coding in Section II. We then proceed to describe our proposed fast mode decision using RF and the corresponding training and validation processes including feature analyses, sample preparation and the hyperparameter tuning process in Section III. Finally, experimental results of our proposed approach are shown in Section IV followed by conclusions drawn in Section V.

II. SCC INTRA CODING

In SCC intra coding, each video frame is divided into coding tree units (CTUs) of 64×64 size. A recursive quad-tree coding structure is applied to each CTU. For each CTU, it has the size of $2N\times2N$ and can be split into four smaller CUs of $N\times N$, namely sub-CUs. This splitting process is repeated recursively until the smallest CU (SCU) size of 8×8 is reached. In SCC, $2N$ can be chosen as 64, 32, 16 or 8. To find the best combination of CU sizes within a CTU, the encoder performs the RDO process that chooses the optimal CU size by comparing the RD cost obtained by the current CU and the sum of RD costs obtained by its four sub-CUs. In other words, for each CU size, the mode that obtains the least RD cost among INTRA, IBC and PLT, is selected as the optimal mode, m^*, as follows:

$$\begin{aligned} &J_m = D_m + \lambda \cdot R_m \\ &m^* = \arg\min_{m\in M}(J_m) \\ &M = \{INTRA, IBC, PLT\} \end{aligned} \qquad (1)$$

where D_m, R_m and J_m, respectively, are the distortion, coding rate and RD cost obtained by the mode m, and λ is the Lagrangian multiplier controlled by the quantization parameter (QP). The CU is then split if the sum of the RD costs of the four sub-CUs, $J_{N,i}$, is smaller than the cost of the current CU, J_{2N}, or otherwise the CU is not split, as in:

$$\begin{cases} \sum_{i=0}^{3} J_{N,i} < J_{2N} & \text{, Split} \\ \sum_{i=0}^{3} J_{N,i} \geq J_{2N} & \text{, Not Split} \end{cases} \qquad (2)$$

where i is the index of sub-CU. As a result, the complexity of encoding a CTU is largely increased in SCC compared with HEVC because there are additional IBC and PLT modes for each CU candidate.

To reduce the coding complexity, the current mode decision process is varied depending on the CU size shown in Fig. 2. If the CU size is 32×32 or smaller, IBC is firstly checked by using a set of BVs including the two last coded BVs as well as the neighbor BVs. If there is distortion, the conventional INTRA is checked. After that, the candidates are checked using the skip/merge mode BV predictors. If skip mode is chosen as the best mode, the encoding process of a CU is finished. Otherwise, if the CU size is 16×16 or smaller, IBC is checked by exhaustive block matching with different PU sizes and strategies depending on the CU size [25,31-32]. At last, if the CU size is 32×32 or smaller, PLT is checked.

To analyze the additional complexity brought by IBC and PLT, we encode the first 100 frames of testing sequences with QPs of 22, 27, 32, and 37 under all intra (AI) configuration, which are the settings recommended by common test conditions (CTC) for SCC [45]. The testing sequences are YUV 4:4:4 sequences which include camera-captured content (CC), animation (ANI), text and graphics with motion (TGM), and mixed (MIX) content. Reference software HEVC Test Model Version 16.12 Screen Content Model Version 8.3 (HM-16.12+SCM-8.3, hereafter called SCM-8.3 for the sake of simplicity) [46] is used. Table I tabulates the Bjøntegaard delta bitrate (BD-rate) [47] and the encoding time difference (ΔTime) of the conventional SCC increased by IBC and PLT compared to SCC with both IBC and PLT disabled. ΔTime is defined as the percentage difference of the encoding time using a new approach, Time_{New}, and the encoding time using the conventional SCC encoder, Time_{Orig}:

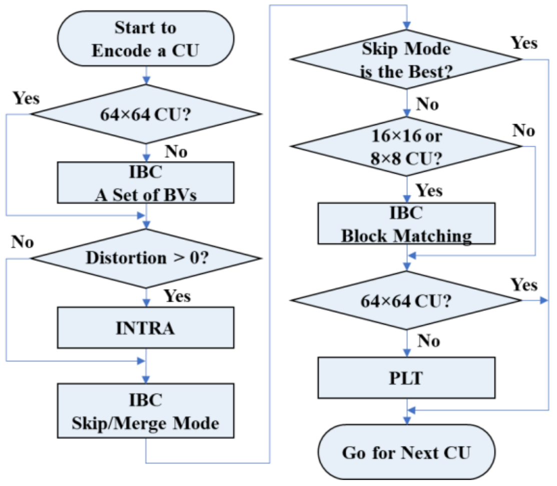

Fig. 2. SCC intra coding process.

TABLE I
BD-RATE (%) AND ΔTIME (%) OF SCC COMPARED TO SCC WITH BOTH IBC AND PLT DISABLED

Sequences	Type	BD-rate	ΔTime
BasketballScreen	MIX	-52.47	87.50
MissionControlClip2	MIX	-48.78	99.70
MissionControlClip3	MIX	-66.92	87.90
ChineseEditing	TGM	-60.68	104.04
Console	TGM	-69.34	52.41
Desktop	TGM	-83.47	66.60
FlyingGraphics	TGM	-63.79	89.28
Map	TGM	-25.87	143.37
Programming	TGM	-52.96	73.50
SlideShow	TGM	-23.38	52.94
WebBrowsing	TGM	-80.32	66.16
Robot	ANI	-2.67	112.42
EBURainFruits	CC	-0.08	91.35
Kimono1	CC	0.03	74.30
Average		**-45.05**	**85.82**

$$\Delta\text{Time} = 100 \times (\text{Time}_{New} - \text{Time}_{Orig})/\text{Time}_{Orig} \qquad (3)$$

It can be observed that the BD-rate is decreased by 45.05% on average and up to 83.47% which indicates that IBC and PLT are efficient coding tools for SC. However, the encoding time is also increased largely by 85.82% on average and up to 143.37% when both IBC and PLT are enabled. Thus, in this paper, a mode skipping approach using RF is proposed to speed up the encoding process of SCC.

III. Proposed Mode Skipping via Random Forest

As aforementioned, the complexity of SCC is largely increased by IBC and PLT though they are efficient coding tools for SC. We therefore propose the mode skipping approach for INTRA, IBC and PLT modes via random forest (RF) so that the number of RD cost computation in (1) can be reduced while still maintaining the coding efficiency. There is only one RF added as a decision block before each mode to be checked. Fig. 3 shows the complete framework of our proposed mode skipping process via RFs. In HEVC, there are four CU sizes from 64×64 to 8×8. Thus, four RFs are required to decide whether IBC skip/merge mode should be skipped for all CU sizes. Similarly, four RFs for INTRA are needed. From Fig.3, since not all CU sizes are used in IBC matching and PLT, two RFs are used in IBC block matching for 16×16 and 8×8 CU

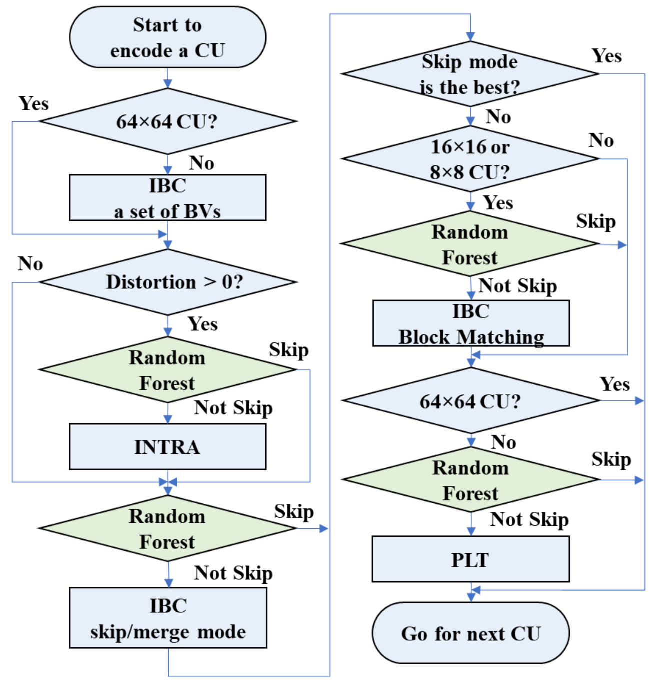

Fig. 3. Our mode skipping approaches via random forests.

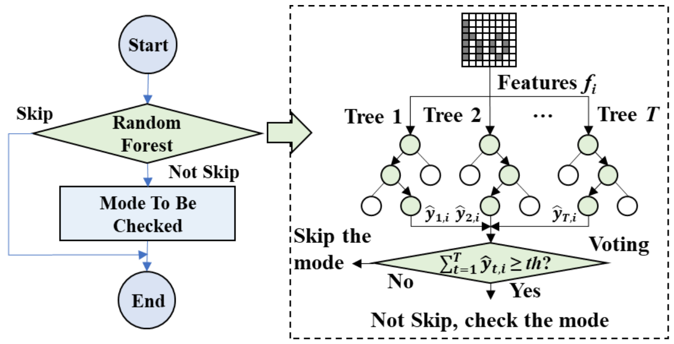

Fig. 4. Proposed mode skipping via random forest.

sizes, while three RFs are used in PLT for 32×32, 16×16 and 8×8 CU sizes. It is noted that there is no RFs prior to the checking of IBC using a set of BVs. This is because within this process, the best BV is determined by the low complexity cost which is the sum of distortion estimated by sum of absolute difference (SAD) computation plus the scaled BV cost by checking the pre-calculated look-up tables. It is relatively computational friendly compared with others. Only the best BV involves the RD computation in (1). Moreover, its RD cost, distortion and rate obtained can be good features for the next RF to check whether the next mode is skipped. By taking these into consideration, we suggest not using RF for deciding whether skip or check this process.

During the coding of the *i*-th CU, for each RF, as in Fig. 4, features $\boldsymbol{f_i}$ are extracted and fed into the trained RF with *T* number of trees. A binary vote, or predicted label, $\hat{y}_{t,i}$ where $\hat{y}_{t,i} \in \{0,1\}$ is the predicted output from *t*-th tree where *t* is the tree index. The predicted outputs of 0 and 1 indicate the mode is voted to be skipped and checked, respectively. If the total sum of $\hat{y}_{t,i}$ is smaller than a threshold *th*, the mode is skipped. Otherwise, it is checked:

$$\sum_{t=1}^{T}\hat{y}_{t,i} \begin{cases} < th & \text{, Skip the mode} \\ \geq th & \text{, Check the mode} \end{cases} \tag{4}$$

In the conventional binary classification problem, *th* is equal to *T*/2 so that the class with majority votes is labelled or classified. We suggest making *th* to be variable such that classification rate can be controllable. For the instance with *th* equal to 1, even if only one tree votes for checking the mode, the mode is checked. This voting makes RF become not greedy which is particularly useful for preserving the video quality during the mode skipping process while achieving complexity reduction to some extent since the mode is skipped only if all trees vote for skipping the mode. We will elaborate more about the tuning of *th* in the hyperparameter tuning in Section III.B.3. There is a special case for 8×8 CU size. When all the modes are decided to be skipped, the mode with the largest number of votes of 1 is checked. If there are more than one modes having the same largest number of votes, those modes are checked.

In the followings, we will describe the features extracted for each CU with analyses. Then, the sample preparation process, which is dedicated to video coding, is presented. Last but not least, the new hyperparameter tuning process with the consideration of bitrate and encoding time will be presented.

A. *Feature Selection*

Extracting good features are the key step to predict the necessary modes being checked and improve the prediction accuracy. Extracting inappropriate features leads to negative influence on the prediction accuracy and increase the computational complexity. Therefore, the feature selection must be related to the characteristics of SC. To have feature analyses, only the first frames of each second from the YUV 4:4:4 sequences in Table I are extracted and encoded with the same setting mentioned in Section II. We will discuss how to deal with RGB and YUV 4:2:0 sequences in Section IV.A.6.

1) *Screen Content Characteristics*

As compared with CC, SC has different characteristics that has complex structure with sharp edges, has only limited number of colors and sometimes has high contrast between colors. Some SC also contain mainly horizontal or vertical edges. The horizontal activity Act_H and vertical activity Act_V are extracted as features as follows:

$$\begin{aligned} Act_H &= \sum_{p_Y \in P}|p_Y(i,j) - p_Y(i-1,j)| \\ Act_V &= \sum_{p_Y \in P}|p_Y(i,j) - p_Y(i,j-1)| \end{aligned} \tag{5}$$

where P is the set of pixels within the CU and $p_Y(i,j)$ is the luminance value at the relative location (i,j) within the CU. Act_H and Act_V are used because SC usually contains sharp edges. These are also the features used in [25] to decide whether the 2D search is employed for IBC, and this fast searching strategy has already been adopted in SCM-8.3 [46]. Boxplots [48] are shown in Fig. 5 for all features related to screen content characteristics used in this paper for 32×32 CU size. More boxplots for other CU sizes can be found in our website [49]. As in Fig. 5(a), the band within the box is the median, i.e. 50th percentile (Q_2), while the bottom and top of the box are the lower quartile (25th percentile, Q_1) and upper quartile (75th percentile, Q_3), respectively. And the range from Q_1 to Q_3 is called interquartile range (IQR). The lower and upper whiskers are the Q_1-1.5×IQR and the Q_3+1.5×IQR, respectively. The remaining points are the outliers. Fig. 5(b) and (c) show the boxplots of Act_H and Act_V for each coding mode, respectively. It can be seen that, for the CUs coded as INTRA, they have much lower Act_H and Act_V values compared with the CUs coded as IBC and PLT. This indicates that Act_H and Act_V are good features to distinguish INTRA and IBC/PLT.

In addition, the variance of a CU is also selected as a feature in the following:

$$Var = \frac{1}{(2N)^2}\sum_{p_Y \in P}\left(p_Y(i,j) - \bar{P}\right)^2 \qquad (6)$$

where $\bar{P}$ is the mean intensity of all pixels in the CU. Variance can help to measure the diversity of pixels within the CU. With larger CU variance, the conventional INTRA is less effective to encode the CU and the chance of using IBC and PLT is larger, which can be shown in Fig. 5(d).

Moreover, the number of high-gradient pixels is estimated as a feature. A pixel is defined as a high-gradient pixel if the luminance difference between itself and one of the neighboring pixels is larger than a pre-defined threshold TH_{HG} as below:

$$\begin{aligned} & p_Y(i,j) \in P_{HG}(TH_{HG}) \text{ if} \\ & \left|p_Y(i,j) - p_Y(i \pm 1, j)\right| > TH_{HG}, \text{ or} \\ & \left|p_Y(i,j) - p_Y(i, j \pm 1)\right| > TH_{HG}. \qquad (7) \\ & N_{HG}(TH_{HG}) = \left|P_{HG}(TH_{HG})\right| \end{aligned}$$

where $P_{HG}(TH_{HG})$ is the set of high-gradient pixels within the CU with different values of TH_{HG}. They are set to 4, 8, 16, and 32 in this paper. The number of elements in $P_{HG}(TH_{HG})$, $|P_{HG}(TH_{HG})|$, are counted as $N_{HG}(TH_{HG})$ which represents the number of high-gradient pixels. $N_{HG}(TH_{HG})$ with different values of TH_{HG} are shown in Fig. 5(e) to (h). From these figures, it can be observed that the range by the box of INTRA has less overlap with the box of PLT as TH_{HG} increases. And there are still CUs coded by IBC when the value of N_{HG} is low. This is because there are SCs with low contrast such as icons, and also there are SCs with high contrast such as texts.

Besides, by concatenating the components of the luminance and chrominance values, i.e. p_Y, p_U and p_V, the number of distinct colors, N_{DC}, is counted since SCs often contain limited number of colors. Higher values of N_{DC} within the CU decreases the chance of choosing PLT as seen in Fig. 5(i) because it probably increases the coding rate of PLT, as demonstrated by Fig. 6(a) which shows coding rate of PLT against N_{DC}. In this figure, it is clearly found that R_{PLT} is small when N_{DC} is low, and vice versa. Also, it can be seen in Fig. 5(i) that IBC is only effective for CUs with small N_{DC} such as texts.

Finally, the number of background colors, N_{BC}, is also considered as a feature by concatenating p_Y, p_U and p_V. The background color is defined to be the most frequently occurred color within the CU. Higher values of N_{BC} within the CU is more likely to be the SC as the SC is computer captured which has no sensor noise. This can be reflected in Fig. 5(j).

2) Neighbor CU Modes

It is logical that when neighbor CUs contain SC, the current CU is likely to be SC. For instance, a word document which contains large amount of texts. Therefore, the modes of left,

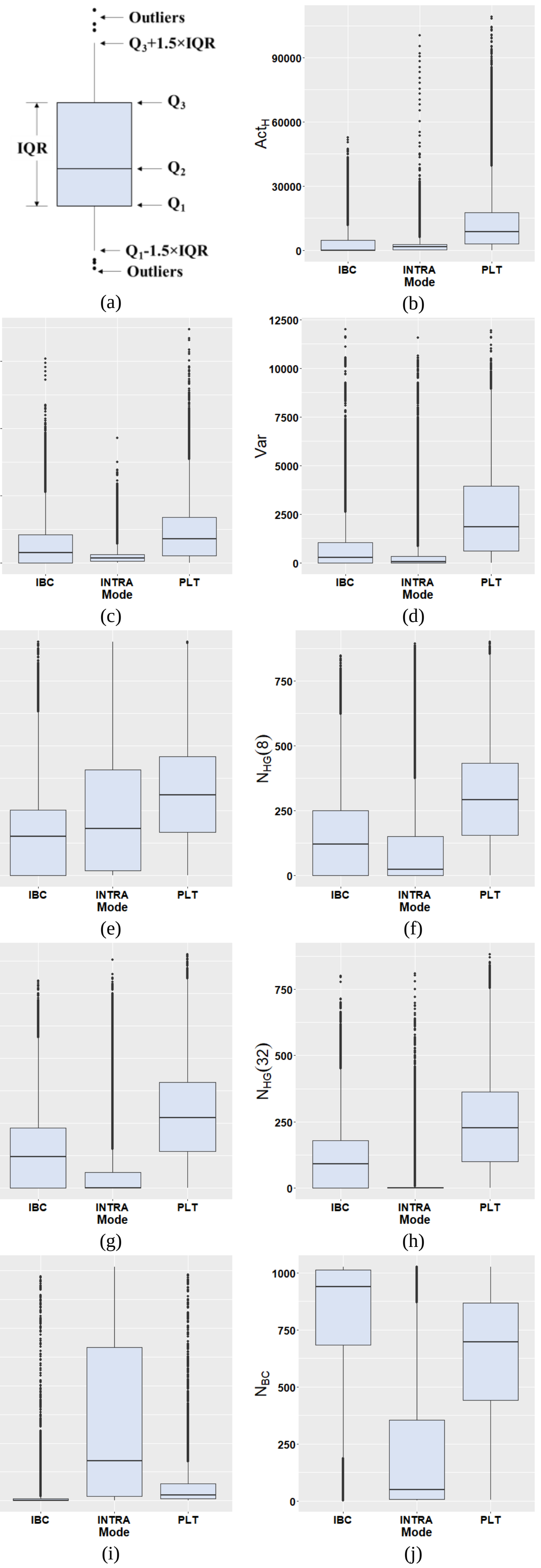

Fig. 5. (a) Illustration of a boxplot and boxplots of screen content characteristics for 32×32 CU size: (b) Act_H, (c) Act_V, (d) *Var*, (e) $N_{HG}(4)$, (f) $N_{HG}(8)$, (g) $N_{HG}(16)$, (h) $N_{HG}(32)$, (i) N_{DC}, and (j) N_{BC}.

TABLE II
AVERAGE NUMBER OF NEIGHBOR CU MODES FOR EACH CODING MODE AT 16×16 CU SIZE

Neighbor CU Modes	Current CU Coding Mode		
	INTRA	**IBC**	**PLT**
Neighbor$_{INTRA}$	**2.74**	0.37	0.70
Neighbor$_{IBC}$	0.39	**3.14**	1.46
Neighbor$_{PLT}$	0.75	0.38	**1.53**
Neighbor$_{NA}$	1.13	1.10	1.32

above, above right, left bottom and above left CUs are extracted to count the numbers of neighbor CU modes that are INTRA, IBC and PLT, i.e. *Neighbor$_{INTRA}$*, *Neighbor$_{IBC}$* and *Neighbor$_{PLT}$*, respectively, as features. Moreover, the neighbor CU location is outside the frame for CUs at the frame boundaries, thus the number of neighbor CU mode that is unavailable, *Neighbor$_{NA}$* is also counted which is useful for CU mode decision at the boundary. The average values of *Neighbor$_{INTRA}$*, *Neighbor$_{IBC}$*, *Neighbor$_{PLT}$* and *Neighbor$_{NA}$* of each coding mode for 16×16 CU size are tabulated in Table II. It is noted that those CUs with other sizes have similar results as Table II which can be found in our website [49]. When the current CU is coded as INTRA, the value of *Neighbor$_{INTRA}$* is 2.74. It means that more than 2 neighbor CUs on average are coded as INTRA when the current CU is also coded as INTRA. This phenomenon also appears in IBC and PLT. For *Neighbor$_{NA}$*, the smallest value of 1.10 is obtained for the current CU coded as IBC. It can be explained that an increase in coding cost of BV happens when the current CU at the left or top boundary in which numerous BV predictors are unavailable. Besides, the search window for IBC becomes restricted which limits the chance of finding repeating patterns. If repeating patterns cannot be found, they are either coded as INTRA and PLT depending on the SC characteristics mentioned in the previous sub-section.

3) *Intermediate Cost Information*

The best mode m_{best} just before the mode being checked or skipped, as well as the corresponding RD cost $J_{m_{best}}$, distortion $D_{m_{best}}$, and the coding rate $R_{m_{best}}$ as in (1), are also selected as features. This is because if the RD cost, distortion or coding rate of the best mode is very small, it is able to give an insight to the encoder that the previously checked modes may be sufficiently effective that the current mode can be skipped. In contrast to CC, the chance of getting the exact or very close match within the same frame of SC is higher resulting in very low $J_{m_{best}}$, $D_{m_{best}}$, and $R_{m_{best}}$. Therefore, we collect the latest intermediate cost information, i.e. the RD cost, distortion and coding rate of the best mode just before the target mode to decide whether it should be skipped or not. For example, if a CU is already well coded by IBC with a very low RD cost, it is likely to skip the following PLT with negligible impact to the coding efficiency.

We further analyze the least RD cost $J_{m_{best}}$ just before the checking of PLT against the number of distinct colors N_{DC} for 32×32 CU size in Fig. 6 (b). 32×32 CU size is chosen for analysis because the number of samples is fewer for the sake of clearer visualization. In Fig. 6(b), with high $J_{m_{best}}$ before the checking of PLT and low N_{DC}, CUs are most likely coded as PLT. This is because high $J_{m_{best}}$ means that the modes checked before PLT cannot encode those CUs efficiently. But with low

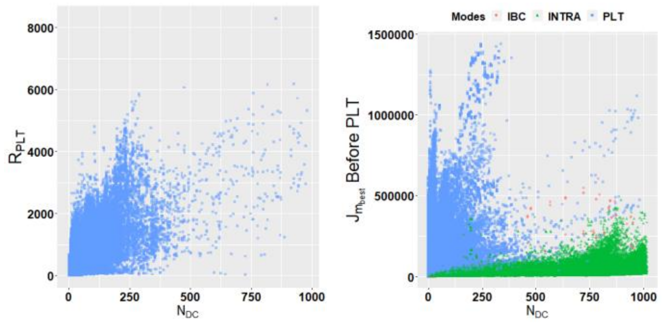

Fig. 6. (a) The coding rate of PLT against N_{DC}, and (b) the least RD cost just before PLT against number of distinct colors, with the modes encoded as INTRA (green triangle), IBC (red circle) and PLT (blue square) for 32×32 CU size.

N_{DC}, those CUs can be encoded by PLT efficiently. We can also see that with low $J_{m_{best}}$ before the checking of PLT, as N_{DC} increases, more CUs are coded as INTRA. It is noted that there are relatively few CUs coded as IBC because repeating patterns tend to be small size and block matching is only performed for 16×16 and 8×8 CU sizes. Thereby, $J_{m_{best}}$ is a useful feature, with the collaboration of other features such as N_{DC}, for deciding whether the mode should be skipped or not.

4) *Feature Vector and Sample Formation*

After extracting the features, a feature vector $\boldsymbol{f}$ is formed, which is input to a RF for mode skipping decision, as follows:

$$\begin{aligned}\boldsymbol{f} = (&Act_H, Act_V, Var, N_{HG}(4), N_{HG}(8), N_{HG}(16),\\ &N_{HG}(32), N_{DC}, N_{BC}, Neighbor_{INTRA}, Neighbor_{IBC},\\ &Neighbor_{PLT}, Neighbor_{NA}, J_{m_{best}}, D_{m_{best}}, R_{m_{best}}, m_{best})\end{aligned} \quad (8)$$

Each sample s_i has its associated feature vector $\boldsymbol{f_i}$ and label y_i as below:

$$s_i = (\boldsymbol{f_i}, y_i) \quad (9)$$

where $y_i \in \{0,1\}$ which is the true label of the *i*-th sample. The value of 1 means the mode of this sample is encoded into the bitstream while the value of 0 means the mode of this sample is not encoded into the bitstream.

B. *Training and Validation*

In this sub-section, firstly, we define the training, validation and testing sets for our proposed mode skipping approach. Secondly, the way to collect and define the samples is discussed. Then, the hyperparameter tuning which is based on the BD-rate and encoding time reduction is described.

1) *Training, Validation and Testing Sets*

To train the RFs with the features mentioned in the previous sub-section, training and validation sets are totally independent of the test set in Table I. The training set are BigDuck, CadWaveform, ChineseDocumentEditing, EBULupo-Candlelight, KristenAndSaraScreen, MissionControlClip1, ParkScene, PcbLayout, RealTimeData, Seeking, VenueVu, Viking, and WordEditing. The validation set is VideoConferencingDocSharing. These sequences are either from JCT-VC [50] or from Joint Video Exploration Team (JVET) for Versatile Video Coding (VVC) [51]. Only the first frames of each second from the sequences are extracted and formed as frame-skipped sequences for both training and

TABLE III
NUMBER OF SAMPLES FOR EACH MODE AT VARIOUS CU SIZES

CU Sizes	INTRA	IBC	PLT
64×64	43052	21524	N/A
32×32	346352	205056	135328
16×16	1013232	1090568	329132
8×8	4086116	4686888	675536

validation sets. After hyperparameter tuning using the training and validation sets, testing process will be performed to evaluate our proposed approach in Section IV.

2) Sample Preparation

The sample preparation for video coding is different from other conventional object classification problems. In details, the training set is required to be encoded by SCM-8.3 [46] using QPs of 22, 27, 32 and 37 with AI configuration which is recommended by CTC [45]. The CUs that are encoded into the bitstream are treated as positive and negative samples according to which mode being skipped. For example, if a RF is to skip INTRA mode, then the CUs which are encoded as INTRA into the bitstream are considered as the positive samples while the CUs which are encoded as IBC or PLT into the bitstream are the negative samples. Also, as the optimal CU size is chosen through the quad-tree encoding process based on (2), there is a sub-optimal mode for each CU size based on (1). Only those with the least RD costs based on (1) and (2) are truly encoded into the bitstream. Therefore, those sub-optimal modes for different CU sizes are treated as negative samples for all modes being skipped. For instance, if the mode is optimal at 32×32 CU size, all the sub-optimal modes obtained at 64×64, 16×16, 8×8 CUs are also treated as the negative samples at those particular CU sizes since at the end they are not encoded into the bitstream. Thus, the CUs consist of mixed SC and CC and may be encoded at smaller CU sizes are also included as the negative samples for training.

As the number of negative samples is always larger than that of positive samples and RFs needs for balanced data, the negative samples are randomly sub-sampled to make the number of negative samples equal to that of positive samples to avoid the data imbalance problem. Table III tabulates the number of samples used for training for each mode at various CU sizes.

3) Hyperparameter Tuning

Our proposed RFs are trained using the random forest package [52] in a free statistical computing language and software called R [53]. Suppose there are *T* number of trees for a RF and d number of features for each RF, $\sqrt{d}$ number of features (with rounded down) at each node of a tree are selected randomly out of *d* features with replacement. Each node is split using the best feature based on the Gini impurity or information gain among that subset of features. The split is terminated if either one of the leaf nodes, i.e. the left and right leaf nodes, has the number of samples smaller than *s* of the total samples to limit the depth of each tree [54-56]. After that, a threshold *th* in (4) is used in the voting to make skipping decision. The selection of these model parameters, including *T*, *s*, and *th*, is optimized by hyperparameter tuning in our training process.

One of the conventional ways for hyperparameter tuning of a RF is to measure the out of bag (OOB) error rate and the one with the lowest OOB error rate is chosen as optimal. For each tree, 63.2%, roughly two-third, of the total training samples *S*, by bootstrap sampling, are input for training. After training, the leftover 36.8% samples, $S_{OOB,t}$, are used for calculating the misclassification rate of *t*-th tree, i.e. the out of bag (OOB) error rate $ER_{OOB,t}$ where *t* is the tree index. And the OOB error rate of a RF $ER_{OOB,RF}$ can be obtained by getting the average OOB error rates from all trees as follows:

$$\begin{aligned} ER_{OOB,t} &= \sum_{i\in S_{OOB,t}} \left|\hat{y}_{t,i} - y_i\right| / \left|S_{OOB,t}\right| \\ ER_{OOB,RF} &= \frac{1}{T}\sum_{t=1}^{T} ER_{\ OOB,t} \end{aligned} \tag{10}$$

where y_i and $\hat{y}_{t,i}$ are the true and predicted labels respectively as in (4), and $\left|S_{OOB,t}\right|$ is the number of samples in the set $S_{OOB,t}$. That is the merit of random feature subspace and random data subset which makes RF not greedy [54-56]. However, the RF trained with the least OOB error rate may not be best fitted for video coding optimization because the OOB error rate only measures the misclassification rate in which it does not well consider the BD-rate and encoding time in SCC. This is because misclassification does not necessarily increase BD-rate. If a mode of a CU should be skipped but wrongly classified by the RF that the mode is going to be checked. In this case, it has no negative impact on BD-rate but it only increases the encoding time. In another case, if a mode of a CU should be optimal and encoded but wrongly classified by the RF that the mode is going to be skipped, but another mode can also be encoded with the RD cost as low as the optimal one, then there is negligible impact on BD-rate but it also helps to decrease the encoding time. This can be happened for homogeneous CUs which have few distinct colors and contain repeating pattern within the same frame that can be efficiently encoded by all of the INTRA, IBC and PLT.

Hence, instead of hyperparameter tuning using OOB error rate, we further propose to tune the hyperparameters *T*, *s* and *th* for our mode skipping approach such that it has the large encoding time reduction with negligible impact on BD-rate. In our approach, RFs are trained using the training set with the number of trees *T* from 7 to 16, and the minimum number of samples in leaf node, i.e. *s* portion of total samples, from 0.05, 0.1, 0.15, 0.2, 0.25 to 0.3 of the total samples. The voting threshold *th*, as in (4), is ranging from 1 to 4. By encoding the validation set, mentioned in Section III.B.1, with different sets of the trained hyperparameters *T*, *s* and *th* for each CU size, the optimal set of hyperparameters $rf_{2N}^{*}(T,s,th)$ is chosen with the minimum increase in BD-rate $BD\text{-}rate_{rf_{2N}(T,s,th)}$ obtained subject to encoding time difference $\Delta\text{Time}_{rf_{2N}(T,s,th)}$ smaller than a target encoding time difference TT_{2N}:

$$\begin{aligned} rf_{2N}^{*}(T,s,th) = \arg \min_{rf_{2N}(T,s,th)\in RF_{2N}} & \left(BD\text{-}rate_{rf_{2N}(T,s,th)}\right) \\ \text{s.t. } \Delta\text{Time}_{rf_{2N}(T,s,th)} & < TT_{2N} \end{aligned} \tag{11}$$

where RF_{2N} is the set of RFs with all possible combinations of hyperparameters. Or alternatively, $rf_{2N}^{*}(T,s,th)$ can be chosen with the maximum encoding time reduction subject to BD-rate smaller than a target BD-rate TR_{2N}:

$$rf_{2N}^{*}(T,s,th) = \arg \min_{rf_{2N}(T,s,th) \in RF_{2N}} \left(\Delta\text{Time}_{rf_{2N}(T,s,th)}\right) \quad \text{s.t. } \textit{BD-rate}_{rf_{2N}(T,s,th)} < TR_{2N} \tag{12}$$

Both (11) and (12) can be used for hyperparameter tuning. In this paper, we select (11) as our preference in order to minimize the increase in BD-rate. Hyperparameter tuning is performed for 64×64 CU size that only RFs at 64×64 CU size are enabled with different sets of T, s and th, and encoded using the validation set to obtain $\textit{BD-rate}_{rf_{64}(T,s,th)}$ and $\Delta\text{Time}_{rf_{64}(T,s,th)}$. Table IV tabulates $\textit{BD-rate}_{rf_{64}}$ and $\Delta\text{Time}_{rf_{64}}$ obtained for 64×64 CU size with T=8 only. More results for other values of T are tabulated in [49]. In Table IV, we can see that with the increase in the value of th, the encoding time reduction is generally larger. This is because with larger th, as in Fig. 4, more votes of 1 from the trees are required in order to check the mode. Different values of T and s are tried so that we can find a set of hyperparameters which gives us a well fitted RF model for speeding up the SCC encoder in terms of BD-rate and encoding time reduction. Based on (11), Table IV and the table in [49] are used to determine $rf_{64}^{*}(T,s,th)$ subject to $\Delta\text{Time}_{rf_{64}(T,s,th)} < TT_{64}$. In principle, the selection of TT_{64} depends on a range of ΔTime with the corresponding reasonable increase in BD-rate resulting in identifying a number of pairs (BD-rate, ΔTime). From Table IV and the table in [49], two pairs of (0.00, -13.07) and (-0.01, -12.88) are likely to be the only reasonable choices. However, it is not necessary to select two ΔTime that are so close. In 64×64 CU size, we therefore only fix at the operational pair of (0.00, -13.07). By taking this into consideration, TT_{64} sets to -13% and rf_{64}^{*}(8,0.15,4) is chosen as the optimal RF model at 64×64 CU size. Table V and the table in [49] tabulate the $\text{BD-rate}_{rf_{32}(T,s,th)}$ and $\Delta\text{Time}_{rf_{32}(T,s,th)}$ for 32×32 CU size on top of rf_{64}^{*}(8,0.15,4) mode skipping approach. The reason of hyperparameter tuning for 32×32 CU size on top of rf_{64}^{*}(8,0.15,4) is just to speed up the hyperparameter tuning process. Results with T=16 are shown in Table V. For 32×32 CU size, we can obtain two operational pairs of ΔTime associated with reasonable BD-rate as: (-0.04, -27.18) and (0.20, -28.02), and TT_{32} can thus be -27% or -28%. For the case of achieving the smallest BD-rate, rf_{32}^{*}(16,0.05,4) is chosen as the optimal RF model at 32×32 CU size by setting TT_{32} to -27%. For 16×16 CU size, based on Table VI and the table in [49], we can choose a range of TT_{2N} from -41%, to -46%, i.e. (0.01, -41.80) and (0.20, -46.70). In between, depends on the granularity, we can have numerous sets of RFs which can have different sets of BD-rate and ΔTime. In our case, two additional sets of RFs (i.e. 4 sets of RFs in total) between -41% and -46% with TT_{2N} equals to -42% and -43% are included, which associates with the operational pairs, (0.07, -42.75) and (0.08, -43.33). Likewise, for 8×8 CU size, based on Table VII and the table in [49], we can choose a range of TT_{2N} from -50%, to -56%. And the corresponding operational pairs are (0.37, -50.91), (0.95, -52.79), (0.96, -54.48) and (1.39, -56.89). Eventually, a mode skipping approach for SCC encoder is adopted by using rf_{64}^{*}(8,0.15,4), rf_{32}^{*}(16,0.05,4), rf_{16}^{*}(12,0.25,4) and rf_{8}^{*}(16,0.25,1) in which the encoder has been well-tuned with the large encoding time reduction and limited impact on BD-rate. Hence, our proposed mode skipping approach with the hyperparameter tuning using (11) is denoted as RF^{*}(TT_{64}, TT_{32}, TT_{16}, TT_{8}). In this case, our optimal mode skipping approach is RF^{*}(-13,-27,-41,-50), which has the minimum increase in BD-rate. Based on the operational pairs that we have just selected, RF^{*}(-13,-27,-42,-52), RF^{*}(-13,-28,-43,-54) and RF^{*}(-13,-28,-44,-56) are the three other possible RFs that can achieve more time reduction with larger increase in BD-rate, as illustrated more in the experimental results. In summary, the selection of TT_{2N} for each $rf_{2N}^{*}(T,s,th)$ depends on (i) which sequence is used as the validation set, (ii) our objective to either preserve the coding efficiency more with less encoding time reduction or sacrifice the coding efficiency for larger reduction of encoding time, and (iii) a range of ΔTime which has the corresponding reasonable increase in BD-rate. In this range, we can choose a small TT_{2N}

TABLE IV
BD-RATE (%) AND ΔTIME (%) OF PROPOSED APPROACH WITH VARIOUS SETS OF HYPERPARAMETERS FOR 64×64 CU COMPARED TO CONVENTIONAL SCC

2*N*=64	***th* = 1**		***th* = 2**		***th* = 3**		***th* = 4**	
***T* = 8**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**
***s* = 0.05**	0.06	-9.67	0.03	-10.47	0.07	-8.51	0.03	-11.38
***s* = 0.1**	0.01	-6.87	0.03	-9.16	0.05	-11.51	0.03	-12.05
***s* = 0.15**	0.01	-5.34	0.01	-11.12	0.01	-12.10	**0.00**	**-13.07**
***s* = 0.2**	0.03	-5.43	0.03	-5.21	0.06	-8.21	0.06	-8.61
***s* = 0.25**	0.00	-5.56	0.04	-8.43	0.01	-8.65	0.06	-5.81
***s* = 0.3**	0.00	-5.66	0.00	-8.63	0.00	-8.51	0.05	-8.79

TABLE V
BD-RATE (%) AND ΔTIME (%) OF PROPOSED APPROACH WITH VARIOUS SETS OF HYPERPARAMETERS FOR 32×32 CU COMPARED TO CONVENTIONAL SCC

2*N*=32	***th* = 1**		***th* = 2**		***th* = 3**		***th* = 4**	
***T* = 16**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**
***s* = 0.05**	0.04	-24.51	-0.04	-25.40	0.05	-26.23	**-0.04**	**-27.18**
***s* = 0.1**	0.13	-22.27	0.13	-24.39	0.12	-25.43	0.10	-25.46
***s* = 0.15**	0.02	-23.30	0.17	-25.24	0.10	-25.84	0.09	-26.38
***s* = 0.2**	0.13	-17.71	0.16	-25.30	0.16	-25.16	0.13	-26.09
***s* = 0.25**	0.09	-26.12	0.09	-25.90	0.06	-25.98	0.01	-26.25
***s* = 0.3**	0.19	-22.88	0.15	-23.62	0.10	-26.20	0.36	-26.76

TABLE VI
BD-RATE (%) AND ΔTIME (%) OF PROPOSED APPROACH WITH VARIOUS SETS OF HYPERPARAMETERS FOR 16×16 CU COMPARED TO CONVENTIONAL SCC

2*N*=16	***th* = 1**		***th* = 2**		***th* = 3**		***th* = 4**	
***T* = 12**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**
***s* = 0.05**	0.03	-33.16	-0.09	-34.40	0.78	-36.24	2.92	-39.43
***s* = 0.1**	0.13	-31.13	0.11	-34.17	0.09	-35.78	-0.02	-36.87
***s* = 0.15**	0.08	-31.80	0.09	-34.66	0.09	-35.38	0.16	-37.23
***s* = 0.2**	0.13	-30.95	0.15	-32.59	0.16	-34.01	0.13	-39.79
***s* = 0.25**	0.09	-31.15	0.08	-32.76	0.01	-39.00	**0.01**	**-41.80**
***s* = 0.3**	0.20	-31.53	0.15	-32.91	0.26	-33.41	0.29	-34.02

TABLE VII
BD-RATE (%) AND ΔTIME (%) OF PROPOSED APPROACH WITH VARIOUS SETS OF HYPERPARAMETERS FOR 8×8 CU COMPARED TO CONVENTIONAL SCC

2*N*=8	***th* = 1**		***th* = 2**		***th* = 3**		***th* = 4**	
***T* = 16**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**
***s* = 0.05**	0.58	-50.83	0.76	-51.05	1.27	-54.78	1.39	-56.89
***s* = 0.1**	0.52	-50.64	0.62	-51.47	0.77	-51.18	1.12	-54.74
***s* = 0.15**	0.67	-50.80	0.92	-51.83	0.93	-51.41	1.06	-54.54
***s* = 0.2**	1.06	-54.41	1.28	-55.72	1.34	-55.74	1.68	-55.89
***s* = 0.25**	**0.37**	**-50.91**	1.17	-54.54	1.32	-54.85	1.49	-55.91
***s* = 0.3**	0.51	-51.44	1.37	-55.27	1.25	-54.80	1.29	-55.83

and a large TT_{2N}. In between, depends on the granularity, we can have numerous sets of RFs which can have different sets of BD-rate and ΔTime.

IV. Experimental Results

To evaluate the performance of the proposed mode skipping approaches using RF, we have performed simulations using the HEVC SCC reference software SCM-8.3 [46] with the CTC [45] mentioned in Section II. The first 100 frames of the 14 sequences in Table I were encoded. The experiments were conducted on the Dell Precision T1700 computer with an Intel i7-4770 3.40GHz processor and 16GB memory. In the followings, we first evaluate the performance of our proposed approaches. Second, our proposed approaches will compare with the state-of-the-art approaches using standard sequences in CTC [45] as well as an extra test set. Moreover, inference time, memory consumption and sequences with different color spaces are also evaluated.

A. Performance Evaluation of Proposed Approaches

1) Hyperparameter Tuning Methods

Since our proposed mode skipping approach can be scalable by choosing different target time TT_{2N} in (11), different sets of BD-rate and encoding time reduction can be achieved. For the sake of simplicity, RF^*(-13,-27,-41,-50), RF^*(-13,-27,-42,-52), RF^*(-13,-28,-43,-54) and RF^*(-13,-28,-44,-56) are denoted as RF_1^*, RF_2^*, RF_3^* and RF_4^*, respectively. In order to show the importance of hyperparameter tuning using (11), we also implemented our mode skipping approach with the hyperparameter tuning using the OOB error rate defined in (10) and it is denoted as $RF_{OOB\text{-}th}$. For $RF_{OOB\text{-}th}$, the hyperparameters T and s are chosen based on the minimum OOB error rate for each RF, and the hyperparameter th is set as 1 to 4. When th=1, among all trees, even only one tree has the vote for checking the mode, the mode is checked. It is the most rate-distortion preserving value. Table VIII shows the average BD-rate and ΔTime of RF_n^* (n = 1 to 4) and $RF_{OOB\text{-}th}$ (th = 1 to 4) against the conventional SCC. We can see that, say for example, between $RF_{OOB\text{-}2}$ and RF_1^* , RF_1^* has larger encoding time reduction of 45.06% and smaller increase in BD-rate of 1.08%, which has better performance than $RF_{OOB\text{-}2}$. Similarly, RF_2^* has larger encoding time reduction of 48.51% while having smaller increase in BD-rate of 1.55% which is better than the performance of $RF_{OOB\text{-}3}$. For better visualization, we also plot the average BD-rate against encoding time reduction in Fig. 7. In Fig. 7, we can see that there is a large margin between the curves of RF_n^* and $RF_{OOB\text{-}th}$. Along the same BD-rate, RF_n^* obtain larger encoding time reduction. This means that our hyperparameter tuning based on the BD-rate and encoding time reduction in (11) is crucial for choosing a set of optimal hyperparameters. It is consistent with our explanation in the previous section that misclassification does not necessarily increase BD-rate largely. If there is misclassification in the sense that a mode is skipped but wrongly classified by the RF that it is being checked, there is no harms on BD-rate but only with an increase in encoding time for this CU. In addition, some of the CUs, such as homogeneous CUs, can be efficiently encoded by multiple modes with similar RD cost. With some modes skipped by RFs, encoding time can be reduced largely with negligible or even zero increase in BD-rate. Thus, our proposed RF_1^* can obtain 45.06% encoding time reduction with only 1.08% increase in BD-rate. And our RF_3^* and RF_4^* have 2.68% and 4.47% increase in BD-rate with 52.51% and 58.57% average encoding time reduction respectively, which shows that our approach is complexity scalable by choosing different target time TT_{2N} in (11).

TABLE VIII
Average BD-rate (%) and ΔTime (%) of Mode Skipping Approach With Different Hyperparameter Tuning Methods Compared To Conventional SCC

Approaches	$RF_{OOB\text{-}1}$	$RF_{OOB\text{-}2}$	$RF_{OOB\text{-}3}$	$RF_{OOB\text{-}4}$	RF_1^*	RF_2^*	RF_3^*	RF_4^*
BD-rate	0.83	1.43	1.89	2.31	1.08	1.55	2.68	4.47
ΔTime	-35.87	-41.29	-45.27	-49.25	-45.06	-48.51	-52.51	-58.57

TABLE IX
Average BD-rate (%) and ΔTime (%) of Mode Skipping Approach With RFs Enabled for Individual Coding Modes Compared To Conventional SCC

Ave-rage	RF_1^* **(Only IBC Skip/Merge)**		RF_1^* **(Only INTRA)**		RF_1^* **(Only IBC Block Matching)**		RF_1^* **(Only PLT)**		RF_1^*	
	BD-rate	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**
MIX	0.28	-16.47	0.65	-18.63	0.37	-8.60	0.06	-8.69	1.57	-44.10
TGM	0.07	-19.93	0.67	-23.85	0.13	-5.91	0.06	-4.84	1.06	-46.46
ANI	0.04	-6.26	0.27	-8.10	0.72	-25.52	0.55	-20.87	1.74	-46.24
CC	0.00	-0.95	0.02	0.38	0.09	-27.11	0.00	-12.28	0.11	-40.33
Overall	0.11	-15.50	0.55	-18.14	0.21	-10.92	0.08	-7.87	1.08	-45.06

TABLE X
Average BD-rate (%) and ΔTime (%) of Mode Skipping Approach With RFs Trained With Different Feature Subsets Compared To Conventional SCC

Average	RF_1^* **(Only SC Characteristics)**		RF_1^* **(Only SC Characteristics and Neighbor CU Modes)**		RF_1^*	
	BD-rate	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**
MIX	1.66	-26.95	1.51	-39.01	1.57	-44.10
TGM	1.66	-34.61	1.25	-37.95	1.06	-46.46
ANI	2.27	-28.95	2.03	-49.72	1.74	-46.24
CC	0.19	-33.23	0.16	-42.35	0.11	-40.33
Overall	1.49	-32.37	1.20	-39.65	1.08	-45.06

2) Individual Coding Modes

We have evaluated the performance of RFs for each coding mode in Table IX. It can be seen that RFs for both IBC skip/merge and INTRA modes contribute more time reduction for MIX and TGM sequences, while RFs for both IBC block matching and PLT modes contribute more time reduction for ANI and CC sequences. It is expected since IBC block matching and PLT modes are designed for SC and these modes are expected to be skipped for ANI and CC sequences. On average, all RFs contribute certain amount of time reduction with negligible impact on BD-rate.

3) Different Feature Subsets

Ablation study on the performance of the RFs trained by different feature subsets are also shown in Table X. With RFs trained only by the features of SC characteristics, 32.37% time reduction with increase in 1.49% BD-rate is obtained. With RFs trained only by the features of SC characteristics and neighbor CU modes, an improved 39.65% time reduction with 1.20% increase in BD-rate is obtained. With RFs trained with

TABLE XI
AVERAGE BD-RATE (%) AND ΔTIME (%) OF MODE SKIPPING APPROACH WITH RFS TRAINED WITH DIFFERENT NUMBER OF HIGH-GRADIENT PIXELS FEATURES COMPARED TO CONVENTIONAL SCC

Average	RF_1^* ($N_{HG}(TH_{HG})$ by (13))		RF_1^* ($N_{HG}(TH_{HG})$ by (7))	
	BD-rate	ΔTime	BD-rate	ΔTime
MIX	1.75	-42.47	1.57	-44.10
TGM	1.22	-45.77	1.06	-46.46
ANI	2.19	-45.00	1.74	-46.24
CC	0.34	-36.73	0.11	-40.33
Overall	1.28	-43.72	1.08	-45.06

intermediate cost information features, an even better time reduction of 45.06% with only 1.08% increase in BD-rate is achieved.

To investigate the $N_{HG}(TH_{HG})$ feature, the counting of $N_{HG}(TH_{HG})$ based on (7) has also been replaced by

$$p_Y(i,j) \in P_{HG}(TH_{HG}) \text{ if}$$
$$\left|p_Y(i,j) \pm p_Y(i \pm 1,j)\right| > TH_{HG}, \text{ or}$$
$$\left|p_Y(i,j) \pm p_Y(i,j \pm 1)\right| > TH_{HG}. \quad (13)$$
$$N_{HG}(TH_{HG}) = |P_{HG}(TH_{HG})|$$

which represents the oblique orientation. Then the RF based on (13) is trained again. Table XI tabulates the performance of the RF trained with (13). 43.72% time reduction with 1.28% increase in BD-rate is obtained, which is a bit worse than RF_1^*. But it still can be considered that they have similar coding performance.

B. Comparison with State-of-the-art Approaches

To have thorough performance evaluation, we compared our proposed mode skipping approach using random forests RF^* with other state-of-the-art approaches [27-32,34,36-37], as discussed in Section I. It should be emphasized that the SCM versions used by [27-32,34,36-37] are different in the sense that there are numerous enhancements, speed-up techniques and codes cleanup in the more updated version used in our proposed approach. Some of the differences are as follows. In some older versions, the BV signaling for IBC was not unified with the one in the conventional inter mode. Only left and above BVs were used as predictors and there was no merge, skip and AMVP modes for IBC. Thus, the BV predictor derivation mechanism is also different from that of the current SCM version. Besides, there was also no conditional checking whether skip mode is the best mode before going into the time-consuming IBC block matching and PLT as shown in Fig. 2. Moreover, $N\times N$ IBC block matching was done after $2N\times N$ block matching while $N\times N$ IBC block matching is eliminated in the current SCM version. In addition, PLT was enabled in 64×64 CU size but it is disabled now due to the occasional usage. Nevertheless, we have re-implemented the approaches of [27-32,34,36-37] in SCM-8.3 for fair comparison.

Fig. 7 and Table XII show the BD-rate and encoding time reduction of various approaches against the conventional SCC. As mentioned in Section I, the state-of-the-art approaches for comparison mainly employs four kinds of speed-up techniques for SCC, as summarized in Table XIII. The first one is the fast mode decision in which mode(s) is(are) skipped instead of

TABLE XII
BD-RATE (%) AND ΔTIME (%) OF VARIOUS APPROACHES COMPARED TO CONVENTIONAL SCC USING CTC SEQUENCES

Sequences	RF_1^*		[27]		[29]		[36]	
	BD-rate	ΔTime	BD-rate	ΔTime	BD-rate	ΔTime	BD-rate	ΔTime
BasketballScreen	1.86	-41.48	1.11	-40.45	1.44	-22.49	1.26	-21.36
MissionControlClip2	1.50	-43.46	1.07	-41.33	1.25	-34.96	2.42	-36.28
MissionControlClip3	1.35	-47.37	1.15	-40.30	1.92	-25.52	1.62	-25.66
ChineseEditing	0.75	-45.26	0.70	-49.81	1.06	-18.89	1.05	-18.73
Console	0.66	-52.57	3.12	-39.32	2.48	-22.61	1.70	-29.05
Desktop	1.11	-57.94	2.33	-46.60	1.57	-21.91	1.73	-26.07
FlyingGraphics	0.86	-46.63	0.68	-7.04	1.59	-19.56	0.87	-22.90
Map	0.44	-25.74	0.58	-35.06	0.56	-16.40	1.51	-17.37
Programming	1.51	-43.28	0.96	-41.62	1.79	-23.33	1.56	-24.01
SlideShow	2.77	-46.53	1.12	-44.21	4.34	-55.43	2.16	-53.79
WebBrowsing	0.38	-53.70	1.91	-50.22	4.06	-24.33	1.14	-27.26
Robot	1.74	-46.24	0.93	-14.08	5.74	-45.97	1.35	-31.61
EBURainFruits	0.17	-42.46	0.60	-16.65	1.83	-47.72	0.92	-27.27
Kimono1	0.05	-38.21	0.13	0.44	1.49	-74.76	1.22	-26.77
Average (Overall)	**1.08**	**-45.06**	**1.17**	**-33.30**	**2.22**	**-32.42**	**1.47**	**-27.72**

TABLE XIII
SPEED-UP TECHNIQUES INVOLVED FOR VARIOUS APPROACHES

Speed-up Techlniques	RF_n^*/ $RF_{OOB\text{-}th}$	[27]	[28]	[29]	[30]	[31]	[32]	[34]	[36]	[37]
Fast Mode Decision	✓	✓		✓	✓	✓		✓	✓	✓
Fast CU Partitioning			✓	✓					✓	✓
Fast INTRA					✓	✓			✓	
Fast IBC		✓					✓			

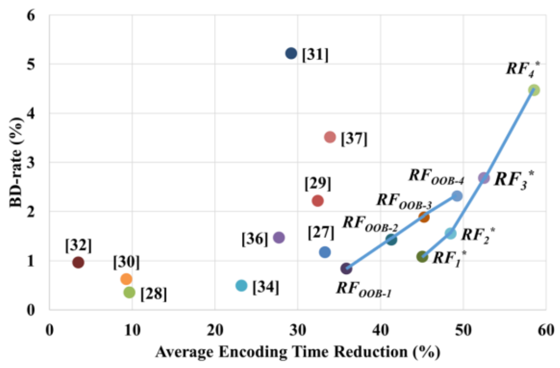

Fig. 7. Comparison of our proposed mode skipping approach via random forests with state-of-the-art approaches with BD-rate (%) against encoding time reduction (%) for the CTC sequences [45].

estimating the RD costs of all modes, as in (1). The second one is the fast CU partitioning. It can be the fast CU splitting method that all of the modes in the current CU size are skipped and the decision process directly moves to the smaller CU size, or it can be the fast CU pruning method that only all modes in the current CU size are performed and the CU does not split further so that all of the modes in the smaller CU size are skipped. Thus, fast CU partitioning is a kind of greedy approach as it skips multiple modes directly. The third one is the fast method within INTRA. The number of INTRA prediction candidates is limited in order to reduce the coding time for checking INTRA. The last one is the fast method within IBC in which certain amount of searching points is skipped such that the time for checking IBC can be reduced. Hence, the last two techniques can obtain the least encoding time reduction.

Fig. 7 depicts the plots of BD-rate against encoding time reduction for different approaches. From Fig. 7, it can be easily seen that the approaches at the right bottom have better coding performance since they have less BD-rate increment and larger encoding time reduction. Our proposed $RF_{OOB\text{-}th}$ (th = 1 to 4) and RF_n^* (n = 1 to 4) are all at the most right bottom in Fig. 7. Particularly for the state-of-the-art approaches with lower than 2% increase in BD-rate, only under 35% encoding time reduction can be obtained. However, our mode skipping approaches RF_1^* and RF_2^* can achieve over 45% of encoding time reduction.

Detailed BD-rate and ΔTime for each sequence are shown in Table XII. In this table, due to the space, only the algorithms in [27, 29, 36] are shown since they have better performance as shown in Fig. 7. Results for other approaches can be found in our website [49]. The algorithms in [28,30] obtain a negligible increase in BD-rate but they only obtain limited average encoding time reduction. This is because their approaches only focuses on homogeneous regions without speed-up consideration for other SC or CC regions. The work in [32] has the least average time reduction in Fig. 7 as it only has the fast hash search within IBC. In Table XII, it can be seen that the algorithm in [29] obtains 32.42% average time reduction but with 2.22% increase in BD-rate which is still a bit large. Both [29] and [31] have large increase in BD-rate. One of the reasons is the use of handcrafted thresholds. And the work in [31] has particularly high BD-rate due to insufficient number of features. The algorithm in [34] gets a small increase in BD-rate but only with 23.22% average time reduction. It is because it needs learning frames for studying the probabilities between features and modes in which there is no time reduction for encoding those learning frames. For [36], the BD-rate and encoding time reduction are much balance that only 1.47% of BD-rate is increased with 27.72% average encoding time reduction. And a significant increase in BD-rate are obtained in [37]. For [29,36-37], they all have a CU type classifier to classify the CU as CCCU or SCCU. If the CU is classified as CCCU, both IBC and PLT are skipped. This may be the reason that the approaches become greedy with the upsurges of BD-rate when there are misclassifications. Thus, the algorithm in [36] suppresses the increase in BD-rate with the sacrifice of smaller encoding time reduction by raising the confidence levels of decision tree classifiers and lowering pre-defined mode skipping thresholds. Among [27-32,34,36-37], [27] has the best coding performance that it only has 1.17% increase in BD-rate and 33.30% average encoding time reduction by encoding the current CU using the mode and CU size from the collocated CU. It mainly expedites the stationary CUs in [27].

Our $RF_{OOB\text{-}1}$ has similar coding performance as [27] with 0.83% increase in BD-rate and 35.87% average encoding time reduction, as shown in Table VIII. With our novel hyperparameter tuning technique using (11), our proposed RF_1^* can obtain 45.06% average encoding time reduction with only 1.08% increase in BD-rate. This can be explained that our mode skipping approach is not greedy in the sense that only one single mode is being skipped, which is decided by RF for each time.

TABLE XIV
BD-RATE (%) AND ΔTIME (%) OF VARIOUS APPROACHES COMPARED TO CONVENTIONAL SCC USING EXTRA TEST SET

Sequences	RF_1^*		[27]		[29]		[36]	
	BD-rate	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**	**BD-rate**	**ΔTime**
JCT-VC Sequences [50]								
CgTwistTunnel	1.18	-41.81	2.01	-9.17	2.44	-26.65	0.71	-24.87
PptDocXls	0.55	-53.66	0.99	-50.08	1.29	-16.66	1.43	-23.65
SocialNetworkMap	0.57	-30.80	0.66	-4.47	0.41	-12.79	0.96	-18.95
JVET Sequences [51]								
BitstreamAnalyzer	0.08	-53.93	0.39	-51.09	0.38	-16.94	0.09	-20.61
CircuitLayout-Presentation	0.60	-47.23	1.02	-43.03	0.93	-19.88	2.50	-20.31
ClearTypeSpreadsheet	0.14	-58.48	0.54	-48.31	0.71	-20.46	1.65	-22.89
EnglishDocument-Editing	0.68	-51.65	3.25	-44.57	1.16	-20.18	1.68	-19.81
Self-Captured Sequences [49]								
MsStore	0.95	-41.60	0.61	-25.35	2.83	-36.30	1.97	-32.41
NewsBrowse	0.50	-51.21	0.68	-38.82	2.57	-37.10	0.97	-35.76
PaperPdf	0.38	-54.85	1.05	-32.01	2.70	-25.36	0.54	-27.15
VisualStudio	0.73	-54.28	1.11	-47.27	2.40	-29.66	1.98	-30.76
YouTube	0.77	-44.18	0.63	-34.51	1.59	-35.91	1.08	-36.38
Average								
JCT-VC	**0.77**	**-42.09**	**1.22**	**-21.24**	**1.38**	**-18.70**	**1.03**	**-22.49**
JVET	**0.37**	**-52.82**	**1.30**	**-46.75**	**0.80**	**-19.36**	**1.48**	**-20.90**
Self-Captured	**0.66**	**-49.22**	**0.81**	**-35.59**	**2.42**	**-32.87**	**1.31**	**-32.49**
Overall	**0.59**	**-48.64**	**1.08**	**-35.72**	**1.62**	**-24.82**	**1.30**	**-26.13**

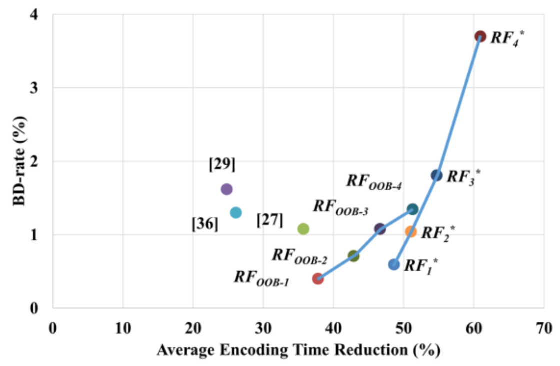

Fig. 8. Comparison of our proposed mode skipping approach via random forests with state-of-the-art approaches with BD-rate (%) against encoding time reduction (%) using extra test set [49-51].

Misclassifications in our approach brings less harmful compared with those using CU type classification in [29,36-37]. Another reason is that while going through the RF as in Fig. 3, the latest intermediate cost information, mentioned in Section IV.A.3, i.e. the RD cost $J_{m_{best}}$, distortion $D_{m_{best}}$, and the coding rate $R_{m_{best}}$, are also considered jointly with the SC characteristics and neighbor CU modes. This information gives more clues to RF whether the mode should be skipped or not. Yet, [29,36] do not consider the intermediate cost information during the CU type classification. It can be concluded that our proposed mode skipping approaches via RFs outperform other state-of-the-art approaches.

C. Performance Evaluation Using Extra Test Set

We also encoded an extra test set which consists of SC sequences from JCT-VC [50], JVET [51] and self-captured sequences [49] as in Table XIV. The self-captured sequences are the screen capture of reading a PDF file, browsing web news, and watching a YouTube video, etc. Fig. 8 shows the

TABLE XV
AVERAGE BD-RATE (%) AND ΔTIME (%) OF MODE SKIPPING APPROACH FOR RGB AND YUV 4:2:0 SEQUENCES

Average	RF_1^* (RGB)		RF_1^* (YUV 4:2:0)	
	BD-rate	ΔTime	BD-rate	ΔTime
MIX	1.43	-40.56	1.55	-33.78
TGM	0.89	-42.12	1.22	-32.13
ANI	1.33	-46.21	1.05	-42.22
CC	0.06	-44.38	-	-
Overall	0.92	-42.40	1.27	-34.06

TABLE XVI
AVERAGE INFERENCE TIME (%) OF MODE SKIPPING APPROACH

RF_1^*	MIX	TGM	ANI	CC	All
	1.76	1.42	3.57	6.86	2.42

TABLE XVII
MEMORY ANALYSIS OF MODE SKIPPING APPROACH

Frame Size (Pixels)	2560×1440	1920×1080	1280×720
Frame Memory for one frame (Bytes)	2560×1440×3	1920×1080×3	1280×720×3
Maximum Additional Memory for 64×64 CU (Bytes)	64×64×36	64×64×36	64×64×36
Memory Increased (%)	1.33	2.37	5.33

average BD-rate against average encoding time reduction of various approaches against the conventional SCC. Our proposed RF_n^* (n = 1 to 4) have the largest encoding time reduction comparing with [27,29,36] at different ranges of BD-rate in which the results are consistent with those in the previous sub-section. From the figure, it is observed that RF_1^* and RF_2^* can even achieve larger reduction on encoding time and smaller increase in BD-rate for these unseen data set. This confirms that our proposed RF_n^* is generalizable to the unseen sequences.

D. RGB and YUV 4:2:0 Color Formats

Instead of training RF again using RGB and YUV 4:2:0 sequences, we use the same set of RFs trained by YUV 4:4:4 sequences. To extract Act_H, Act_V, Var and $N_{HG}(TH_{HG})$ features based on (5) to (7), sequences in RGB are converted to YUV 4:4:4 format to obtain the Y component. When N_{DC} and N_{BC} are determined, three components of R, G, and B are concatenated to get a 24-bit sample value for each pixel. For YUV 4:2:0 sequences, the nearest smaller integer positions of U and V are picked for concatenation when N_{DC} and N_{BC} are estimated. Table XV shows the performance of RGB and YUV 4:2:0 sequences. The RGB and YUV 4:2:0 sequences are exactly following the CTC (Noted that there are no CC sequences for YUV 4:2:0 in CTC) [45]. 42.40% and 34.06% time reduction with 0.92% and 1.27% increase in BD-rate are obtained for RGB and YUV 4:2:0 sequences respectively. The time reduction is relatively smaller for YUV 4:2:0 sequences because there are many coding techniques disabled such as cross component prediction and adaptive color transform. From the table, it is concluded that our approach is generalizable to other color formats.

E. Inference Time, Memory Consumption and Model Size

Table XVI and Table XVII tabulate the inference time and the additional memory analyses of our approach, respectively. The inference time includes the time for feature extraction and decision inference. It is noted that this inference time have been counted in all simulations to calculate the encoding time reduction in the above sections. Though it is relatively large of 6.86% for CC sequences, the average inference time for all sequences is only 2.42% of the total encoding time. With such large encoding time reduction, it is acceptable. For the additional memory, there is only one large memory consumption for the concatenation of color components during the feature extraction processes of N_{DC} and N_{BC}. The memory consumed is the largest when the current CU is 64×64, which is 64×64×36 bytes, i.e. 144 Kbytes only. Compared with the memory to store one frame, it is negligible. Regarding the RF model size, after training RFs using R [53], we implemented the RFs into SCM-8.3 [46] using C++ directly. The original SCM-8.3 has the size of 1.65MB while our RF_1^* to RF_4^* have the sizes of about 1.89MB to 1.91MB. Thus, the RF models increase the size by about 0.24MB to 0.26MB only that is in favor of product implementation. It is noted that most of the RF codes are mainly a series of if-else conditional statements without any special optimizations. In addition, there is no parallel execution on RFs for the sake of fair comparison.

V. CONCLUSION

Screen content coding essentially improves the coding efficiency of screen content by introducing two new tools: IBC and PLT modes. However, they also bring the high computational complexity to the encoder. Therefore, in this paper, we propose the mode skipping approach to skip the conventional INTRA, IBC and PLT modes via RFs. Instead of classifying CU into CCCUs and SCCUs, before checking each mode, feature extraction and classification using RF is performed to decide whether the current mode should be skipped or not. This arrangement allows the latest CU information such as the least rate-distortion cost, the associated distortion and coding rate before the current mode can be used as the input to RF with other features for mode skipping decision. By taking the bitrate and encoding time into consideration, a new hyperparameter tuning process is designed for RFs at each CU size to further enhance the coding efficiency, which is validated by implementing our proposed approach in SCM-8.3. Experimental results have shown that the proposed approach can achieve 45.06% average time reduction with only 1.08% increase in bitrate, which outperforms all of the state-of-the-art algorithms in the literature.

ACKNOWLEDGMENT

We would like to thank the authors from [27,36-37] for the provision of source codes, and would also like to thank Dr. Jun-Jie Huang for the valuable discussion and comments.

REFERENCES

[1] Y. Lu, S. Li, and H. Shen, “Virtualized Screen: A Third Element for Cloud-Mobile Convergence,” *IEEE Multimedia*, vol. 18, no. 2, pp. 4–11, Feb. 2011.
[2] “Joint Call for Proposals for Coding of Screen Content,” *ISO/IEC JTC1/SC29/WG11*, N14175, San Jose, U.S.A., Jan. 2014.
[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. 2015.

[4] W. H. Peng, F. G. Walls, R. A. Cohen, J. Xu, J. Ostermann, A. Maclnnis, and T. Lin, "Overview of Screen Content Video Coding: Technologies, Standards, and Beyond," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 393-408, Dec. 2016.

[5] G. J. Sullvian, J. 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.

[6] G. J. Sullivan, J. M. Boyce, Y. Chen, J. R. Ohm, C. A. Segall, and A. Vetro, "Standardized Extensions of High Efficiency Video Coding (HEVC)," *IEEE J. Sel. Topics. Signal Process*, vol. 7, no. 6, pp. 1001-1016, Dec. 2013.

[7] T. Lin, Peijun Zhang, Shuhui Wang, Kailun Zhou, and Xianyi Chen, "Mixed Chroma Sampling-Rate High Efficiency Video Coding for Full-Chroma Screen Content," *IEEE Trans. Circuits Syst. Video Technol.*, vol.23, no.1, pp. 173-185, Jan. 2013.

[8] S. Wang, K. Gu, S. Ma, and W. Gao, "Joint Chroma Downsampling and Upsampling for Screen Content Image," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1595-1609, Sep. 2015.

[9] 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.

[10] Y. Piao, J. H. Min, and J. Chen, "Encoder Improvement of Unified Intra Prediction," *JCT-VC*, JCTVC-C207, pp. 1-5, Guangzhou, China, Oct. 2010.

[11] 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. Emerging Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 420-432, Dec. 2016.

[12] 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. Emerging Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 409-419, Dec. 2016.

[13] Y. J. Chang, C. C. Lin, C. L. Lin, and P. H. Tsai, "Bi-Color Coding for Screen Visual Content," in *Proc. of Asia-Pacific Signal and Info. Process. Assoc. Annu. Summit and Conf. (APSIPA ASC)*, pp. 1-5, Siem Reap, Cambodia, Dec. 2014.

[14] Y. C. Sun, T. D. Chuang, J. Kim, Y. W. Chen, S. Liu, Y. W. Huang, and S. Lei, "Improved Palette Index Map Coding on HEVC SCC," in *Proc. of IEEE Int. Conf. on Image Process. (ICIP)*, pp. 4210-4214, Phoenix, Arizona, U.S.A., Sep. 2016.

[15] W. Zhu, K. Zhang, J. An, H. Huang, Y. C. Sun, Y. W. Huang, S. Lei, "Inter-Palette Coding in Screen Content Coding," *IEEE Trans. Broadcasting*, vol. 63, no. 4, pp. 673-679, Dec. 2017.

[16] 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. of Int. Conf. on Digital Signal Process. (DSP)*, pp. 888-892, Hong Kong, China, Aug. 2014.

[17] C. C. Chen, and W. H. Peng, "Intra Line Copy for HEVC Screen Content Intra-Picture Prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 7, pp. 1568-1579, Jul. 2017.

[18] J. Lainema, "Intra Block Copy Masking," in *Proc. of, IEEE Int. Conf. on Consum. Electron.*, pp. 333-336, Las Vegas, Nevada, U.S.A., Jan. 2015.

[19] Z. Zhang, and V. Sze, "Rotate Intra Block Copy for Still Image Coding," in *Proc. of IEEE Int. Conf. on Image Process. (ICIP)*, pp. 4102-4106, Quebec City, Canada, Sep. 2015.

[20] K. Zhang, J. An, X. Zhang, H. Huang, and S. Lei, "Symmetric Intra Block Copy in Video Coding," in Proc. of *IEEE Int. Symp. on Circuits Syst. (ISCAS)*, pp. 521-524, Lisbon, Portugal, May 2015.

[21] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Decoder Side Merge Mode and AMVP in HEVC Screen Content Coding," in *Proc. of Int. Conf. on Image Process. (ICIP)*, pp. 260-264, Beijing, China, Sep. 2017.

[22] L. Zhao, T. Lin, K. Zhou, S. Wang, and X. Chen, "Pseudo 2D String Matching Technique for High Efficiency Screen Content Coding," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 339-350, Mar. 2016.

[23] L. Zhao, K. Zhou, J. Guo, S. Wang, and T. Lin, "A Universal String Matching Approach to Screen Content Coding," *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 796-809, Apr. 2018.

[24] M. Budagavi, and D.-K. Kwon, "Intra Motion Compensation and Entropy Coding Improvements for HEVC Screen Content Coding," in *Proc. of Picture Coding Symp. (PCS)*, pp. 365-368, San Jose, California, U.S.A., Dec. 2013.

[25] M. Budagavi, and D.-K. Kwon, "Fast Intra Block Copy (IntraBC) Search for HEVC Screen Content Coding," in *Proc. of IEEE Int. Symp. on Circuits Syst. (ISCAS)*, pp. 9-12, Melbourne, Australia, Jun. 2014.

[26] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, "Hash Based Block Matching for Screen Content Coding," *IEEE Trans. Multimedia*, vol. 17, no. 7, pp. 935-944, Jul. 2015.

[27] 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. of IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, pp. 1377-1381, Shanghai, China, Mar. 2016.

[28] M. Zhang, Y. Guo, and H. Bai, "Fast Intra Partition Algorithm for HEVC Screen Content Coding," in *Proc. of IEEE Visual Comm. Image Process. Conf. (VCIP)*, pp. 390-393, Valletta, Malta, Dec. 2014.

[29] 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. Broadcasting*, vol. 63, no. 1, pp. 48-58, Mar. 2017.

[30] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Fast and Efficient Intra Coding Technique for Smooth Regions in Screen Content Coding Based on Boundary Prediction Samples," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, pp. 1409-1413, Brisbane, Australia, Apr. 2015.

[31] S.-H. Tsang, W. Kuang, Y.-L. Chan, and W.-C. Siu, "Fast HEVC Screen Content Coding By Skipping Unnecessary Checking of Intra Block Copy Mode Based on CU Activity and Gradient," in *Proc. of Asia-Pacific Signal and Info. Process. Assoc. Annu. Summit and Conf. (APSIPA ASC)*, pp. 1-5, Jeju, Korea, Dec. 2016.

[32] 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. of Asia-Pacific Signal and Info. Process. Assoc. Annu. Summit and Conf. (APSIPA ASC)*, pp. 396-400, Hong Kong, China, Dec. 2015.

[33] Y. Kawakami, Gaoxing Chen, and T. Ikenaga, "Content Based Mode and Depth Skipping With Sharp and Directional Edges for Intra Prediction in Screen Content Coding," in Proc. of *IEEE Int. Colloq. on Signal Process. & Its Applicat. (CSPA)*, pp. 46-49, Malacca City, Malaysia, Mar. 2016.

[34] W. Kuang, S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Fast Mode Decision Algorithm for HEVC Screen Content Intra Coding," in *Proc. of Int. Conf. on Image Process. (ICIP)*, pp. 2473-2477, Beijing, China, Sep. 2017.

[35] F. Duanmu, Z. Ma, and Y. Wang, "Fast CU Partition Decision Using Machine Learning for Screen Content Compression," in *Proc. of IEEE Int. Conf. on Image Process. (ICIP)*, pp. 4972-4976, Quebec City, Canada, Sep. 2015.

[36] F. Duanmu, Z. Ma, and Y. Wang, "Fast Mode and Partition Decision Using Machine Learning for Intra-Frame Coding in HEVC Screen Content Coding Extension," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 517-531, Dec. 2016.

[37] H. Yang, L. Shen, and P. An, "An Efficient Intra Coding Algorithm Based on Statistical Learning for Screen Content Coding," in *Proc. of Int. Conf. on Image Process. (ICIP)*, pp. 2468-2472, Beijing, China, Sep. 2017.

[38] 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.

[39] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Fast Intraprediction for High-Efficiency Video Coding Screen Content Coding by Content Analysis and Dynamic Thresholding," *J. Electron. Imaging*, vol. 27, no. 5, pp. 053029-1-053029-18, Oct. 2018.

[40] 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.*, Early Access Article, pp. 1-15, Mar. 2019.

[41] Z.-S. Liu, W.-C. Siu, and J.-J. Huang, "Image Super-Resolution Via Weighted Random Forest," in *Proc. of Int. Conf. on Ind. Technol. (ICIT)*, pp. 1019-1023, Toronto, Canada, Mar. 2017.

[42] J.-J. Huang, W.-C. Siu, and T.-R. Liu, "Fast Image Interpolation via Random Forests," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3232-3245, Oct. 2015.

[43] S. Fang, R. Jin, and Y. Cao, "Fast Depth Estimation From Single Image Using Structured Forest," in *Proc. of IEEE Int. Conf. on Image Process. (ICIP)*, pp. 4022-4026, Phoenix, Arizona, U.S.A., Sep. 2016.

[44] B. Du, W.-C. Siu, and X. Yang, "Fast CU Partition Strategy for HEVC Intra-Frame Coding Using Learning Approach Via Random Forests in *Proc. of Asia-Pacific Signal and Info. Process. Assoc. Annu. Summit and Conf. (APSIPA ASC)*, pp. 1085-1090, Hong Kong, China, Dec. 2015.

[45] "Common Test Conditions for Screen Content Coding," *JCT-VC*, JCTVC-X1015, Geneva, Switzerland, pp. 1-8, May-Jun. 2016.

[46] HEVC Test Model Version 16.12 Screen Content Model (SCM) Version 8.3, HM-16.12+SCM-8.3, [Online], available at:

https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.12+SCM-8.3/, accessed Jun. 2018.
[47] G. Bjøntegaard, “Calculation of Average PSNR Differences Between RD Curves,” *VCEG*, VCEG-M33, Austin, Texas, U.S.A., pp. 1-4, Apr. 2001.
[48] M. Krzywinski, and N. Altman, “Visualizing samples with box plots,” *J. Nature Methods*, vol. 11, no. 2, pp. 119-120. Feb. 2014.
[49] Mode Skipping for HEVC Screen Content Coding via Random Forest. [Online]. Available at: http://www.eie.polyu.edu.hk/~ylchan/research/rfscc/, accessed Jun. 2018.
[50] JCT-VC Screen Content Sequences. [Online]. Available at: ftp://ftp.tnt.uni-hannover.de/testsequences/, accessed Jun. 2018.
[51] J. Guo, L. Zhao, T. Lin, and H. Yu, “Response to B1002 Call for Test Materials: Five Test Sequences for Screen Content Video Coding,” *JVET*, JVET-C0044, pp. 1-9, Geneva, Switzerland, May 2016.
[52] A. Liaw, and M. Wiener, “Classification and Regression by randomForest,” *R News*, vol. 2, no. 3, pp. 18-22, Dec. 2002.
[53] R: The R Project for Statistical Computing. [Online]. Available at: https://www.r-project.org/, accessed Jun. 2018.
[54] G. Louppe, “Understanding Random Forests: From Theory to Practice,” *Ph.D. Dissertation*, University of Liege, pp. 1-211, Jul. 2014.
[55] Tin Kam Ho, “The Random Subspace Method for Constructing Decision Forests,” in *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 20, no. 8, pp. 832-844, Aug 1998.
[56] L. Breiman, “Random Forests”, *J. Mach. Learn.*, vol. 45, no. 1, pp. 5-32, Oct. 2001.

Sik-Ho Tsang (M’10) received the Ph.D. degree from The Hong Kong Polytechnic University (PolyU), Hong Kong, in 2013.

He was a Postdoctoral Fellow from 2013 to 2016, and involved numerous industrial projects for video coding and transcoding. He is currently a Research Fellow in PolyU. He has authored numerous international journals, conferences and patents. His current research fields involve image/video coding such as HEVC, VVC, multiview video plus depth coding, screen content coding, and immersive video coding including light field coding and 360-degree video coding. His research interests also include machine learning and deep learning.

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

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 120 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-degree omnidirectional video coding.

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

Wei Kuang (S’17) received the B. S. degree in School of Electronic and Optical Engineering from Nanjing University of Science and Technology, Nanjing, China, in 2015. Now, he is currently pursuing the Ph.D. Degree in the Department of Electronic and Information Engineering at 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 of international journals including the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS.