SVM based Approach for Complexity Control of HEVC Intra Coding

Farhad Pakdaman[a,b], Li Yu[c], Mahmoud Reza Hashemi[b,*], Mohammad Ghanbari[b,d], and Moncef Gabbouj[a]

[a] *Department of Computing Sciences, Tampere University, Tampere, Finland, P.O. Box 553*
[b] *School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran, P.O. Box: 14395-515*
[c] *Nanjing University of Information Science and Technology, Nanjing, China*
[d] *School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK, CO4 3SQ*

***Abstract*—The High Efficiency Video Coding (HEVC) is adopted by various video applications in recent years. Because of its high computational demand, controlling the complexity of HEVC is of paramount importance to appeal to the varying requirements in many applications, including power-constrained video coding, video streaming, and cloud gaming. Most of the existing complexity control methods are only capable of considering a subset of the decision space, which leads to low coding efficiency. While the efficiency of machine learning methods such as Support Vector Machines (SVM) can be employed for higher precision decision making, the current SVM-based techniques for HEVC provide a fixed decision boundary which results in different coding complexities for different video content. Although this might be suitable for complexity reduction, it is not acceptable for complexity control. This paper proposes an adjustable classification approach for Coding Unit (CU) partitioning, which addresses the mentioned problems of complexity control. Firstly, a novel set of features for fast CU partitioning is designed using image processing techniques. Then, a flexible classification method based on SVM is proposed to model the CU partitioning problem. This approach allows adjusting the performance-complexity trade-off, even after the training phase. Using this model, and a novel adaptive thresholding technique, an algorithm is presented to deliver video encoding within the target coding complexity, while maximizing the coding efficiency. Experimental results justify the superiority of this method over the state-of-the-art methods, with target complexities ranging from 20% to 100%.**

***Keywords*—HEVC, complexity control, intra coding, SVM, machine learning, video compression.**

1. Introduction

The increasing demand for higher quality video content creates a continuous challenge for video networking and storage. The bitrate of higher quality video tends to increase faster than the speed of network technology evolving. Consequently, more efficient video compression standards have been introduced to enable video transmission for High Definition (HD) content. The current High Efficiency Video Coding (HEVC) standard [1] provides almost twice the coding efficiency of the previous standard, H.264/AVC [2]. However, its baseline coding algorithms are much more complex, which makes it challenging for many real time applications [3][4].

Complexity control is necessary for several video applications, such as power-constraint video coding, video streaming, or cloud-gaming. Not only are these applications sensitive to the coding delay/complexity, but also their tolerance to delay/complexity changes dynamically according to the available processing power [5], network conditions [6], or gaming schemes [6][7]. While complexity reduction methods aim to keep the coding efficiency close to its maximum when reducing the complexity, complexity control aims to deliver the encoding at the exact target complexity, while trying to perform the best compression within that processing quota [8][9].

The existing complexity control methods for HEVC either consider early termination of CU partitioning [8], or consider a set of predefined decisions [4][10] to regulate the complexity. These approaches cannot fully exploit the processing power for coding efficiency. Using the capabilities of machine learning techniques to decide the partitioning at each coding depth is an effective way of benefiting from the wide decision space. Recently, the potentials of Support Vector Machines (SVM) have been used for *complexity reduction* in HEVC encoding [3][11]. However, as the existing SVM-based approaches aim to minimize the loss of quality while reducing complexity, they offer different complexity reductions depending on video content. Moreover, due to their fixed decision boundary, they cannot be used to adjust the complexity at the encoding time, i.e. *controlling the complexity*.

As the complexity of intra coding has specifically increased in HEVC, extra efforts have been dedicated to reducing [3][11], and controlling [4][12] its complexity. All-intra coding is useful for video surveillance in wireless sensor networks [13], video archiving [14], screen content coding [15], and video communication in complex networking environment [16].

This paper proposes a method for the complexity control of HEVC intra coding. To exploit the available processing power for the best coding efficiency, an SVM-based approach is employed to decide the CU partitioning at each coding depth. First, a set of features are designed to measure the texture characteristics for CU partitioning. The application of texture analysis via complex wavelet transform and planar filter for fast

* Corresponding author.
Email Address: rhashemi@ut.ac.ir (Mahmoud Reza Hashemi)

Moncef Gabbouj is a member of EURASIP

intra direction estimation has recently been introduced in [17] and [18]. To be able to solve the more challenging problem of CU partitioning and complexity control, this concept is greatly extended in this paper to measure different aspects of texture complexity. Second, to solve the problem of fixed decision boundary in existing SVM-based methods, an adjustable classification approach is developed that performs the classification based on the probability of each class. This approach provides the ability to adjust the complexity-efficiency trade off on the fly. Finally, the coding complexity and loss of coding efficiency are modeled based on the distribution of classification probabilities, and the best partitioning decision that satisfies the target complexity is derived through solving an optimization problem. The main contributions of the paper are summarized as follows:

1) A new hand-crafted feature set for CU partitioning is proposed through analyzing sub-bands of a complex wavelet transform [19]. The features (which are essentially different from the ones used in [17]) are shown to be more effective than traditional features used in the state-of-the-art methods.
2) CU partitioning is modeled through an offline trained SVM-based approach and online distribution modeling. Then, an adaptive thresholding method is proposed to decide the CU partitioning at each coding depth. Compared to the conventional methods which use a fixed binary classification, this approach provides flexibility and adjustable encoding complexity.
3) The relationship between the coding complexity and the partitioning decision is investigated at each coding depth. Accordingly, the encoding complexity (equivalent to encoding time throughout this paper) is modeled, based on the CU partitioning decision. This provides a more accurate model of encoding complexity compared to alternative approaches, which simplifies the CU partitioning problem into a depth estimation problem.
4) The complexity control problem is modeled and solved as a constrained optimization problem. This optimization guarantees the target complexity, while maximizing the coding efficiency, through a content-adaptive process.

The rest of the paper is organized as follows. Section 2 reviews the related works. In section 3, some observations from the HEVC intra coding are presented. The proposed method is presented in section 4. Section 5 presents the experimental results and comparisons with the state-of-the-art methods. Finally, section 6, concludes the paper.

2. Related Works

Several research papers have addressed the high complexity issue of HEVC intra coding. These works can be classified into three major categories.

2.1 Fast mode decision

HEVC offers 35 intra modes for each CU size. To speed-up the intra prediction process, [20][21][22] use information from the encoded neighboring blocks. Intra mode decision in these papers is formulated through the RD cost of the coded blocks, and exploiting the spatial correlations among the blocks.

Texture analysis for fast intra prediction is another major approach which was investigated in [17][18][23][24]. In these papers, signal processing tools such as Sobel and Prewitt operators are adopted to estimate the best intra direction. In our previous works, we have designed features using sub-bands of a Dual-Tree Complex Wavelet Transform (DT-CWT) [19][17], and the Planar filter [18], for fast intra direction prediction. However, these features are designed to estimate the edge directions for fast intra mode selection, not CU partitioning; hence, they are not directly related to this paper. Section 4.1 explains how this paper uses DT-CWT to measure texture complexity, edge complexity, and homogeneity, for CU partitioning.

2.2 Fast CU partitioning

While fast mode decision algorithms try to reduce the number of intra modes checked in each CU size, fast CU partitioning schemes reduce the complexity by minimizing the number of candidate CU sizes that are checked. Usman *et al.* [25] and Min *et al.* [26] exploit the correlation between texture complexity and partitioning decision. To this end, the texture complexity is measured through the pixels' variance in [25], the energy of dominant edges of the block in [3], or by comparing the global and local energy as in [26]. After estimating the texture energy, thresholding techniques are applied to either skip a certain CU size, or to terminate the splitting process.

Another approach to decide the best partitioning, is to exploit the information from neighboring coded blocks [27][28][29]. Cho *et al.* [27] and Lim *et al.* [28] analyze the distribution of the RD cost through the scene. Then a Bayesian decision process is performed to find the best partitioning. A similar method in [30] proposes early termination decision based on a Bayesian model. Kim *et al.* [31] present a two-step approach, where the misprediction penalty is learned through an offline learning process first. Then the Bayesian rule is used to jointly consider the splitting/skipping probabilities, and misprediction costs, to decide the best partitioning. Moreover, Shen *et al.* measure the texture complexity based on the mean absolute deviation of the luma values [32]. If the CU is considered complex, the most similar neighboring blocks are used to decide the most suitable coding depth.

While deep neural network-based methods have been investigated in [33] and [34], other machine learning approaches are more popular in improving the decision accuracy. Lee *et al.* [35] use Fisher's linear discriminant analysis to project features to a more separable space. Then the K-nearest neighbors (K-NN) is used to decide the partitioning. Zhu *et al.* [36] and Grellert *et al.* [37] adopt SVM with a set of encoder-level features, such as prediction modes, depth information and coding bits. [37] sets the SVM's decision threshold to favor CU splitting. However, this is done statically and does not correspond to an adaptive decision making. Zhang *et al.* [11] propose features based on similarity between the neighboring blocks, and train SVMs to decide the CU splitting or CU termination. Liu *et al.* [3] however, present a set of features based on spatial domain image processing techniques. Then, they present a dual-SVM technique that enables a trade-

off between classification accuracy and time saving. However, this trade-off can only be adjusted during the (offline) training time, thus it cannot be used to regulate the complexity during the encoding time. In section 4.2, we present our novel SVM approach that provides this flexibility during the encoding time.

2.3 Controlling the complexity

The performance of fast CU partitioning schemes strongly depends on the video content. As a result, they provide different time savings for different video scenes. However, in many video applications, e.g. power constrained video coding and video networking, the encoder should be able to deliver video compression at a specified target complexity (time or power).

To accurately control the complexity, Correa *et al.* constrain the maximum coding depth for a portion of frames [5][38]. The number of constrained frames and the maximum depths are set according to the target complexity and the outcome of previously unconstrained coded frames, respectively. Another method in [8] considers early termination conditions for complexity control. Offline trained controller schemes are proposed in [10][39], where the best configurations for each complexity level is learned through exhaustive Pareto-frontier analysis. At the encoding time, the next configuration is derived, through a predefined table, based on the current measured complexity and the target complexity.

Deng *et al.* [9] also control the complexity through maximum coding depth, while emphasizing visual quality. First, the encoding complexity and the visual distortion are modeled based on the maximum depth. Then, the coding complexity is distributed among the Coding Tree Units (CTU), based on a visual attention model, to ensure the best visual quality. An objective map has been employed to extend this work in [40].

Zhang *et al.* propose a complexity control scheme for intra coding [4]. The coding complexity of each CTU is estimated using the prediction performance. Then the complexity budget is distributed among CTUs, based on the estimated complexity and predicting the dominant block size. The complexity allocation is performed through finding a subset of candidate CU sizes for each CTU. A similar approach has also been used for inter prediction [41]. Because of single metric for modeling the complexity, and predefined subsets of decision space, this approach leads to sub-optimum decision making. Finally, a time and energy estimation model for intra coding is presented in [12] that can be used for complexity control.

While methods presented in 2.1 and 2.2 can reduce the coding complexity of HEVC encoding, they provide content-dependent solutions and their coding complexity depends on the video sequence. Hence, these methods are not suitable for applications where complexity needs to be adjusted dynamically, i.e. complexity control. Methods in 2.3 aim to control the complexity; however, they mainly provide sub-optimum solutions due to 1) single-feature, or inadequate texture complexity criteria, 2) simplifying the CU partitioning decision into a CTU-level depth estimation, which limits the decision space, or 3) oversimplification due to providing fixed offline solutions. The proposed method in section 4 tackles these issues by designing a feature set that models different aspects of CU partitioning, and a complexity controller, that considers the CU partitioning at each CU depth, leading to the maximum decision space and hence better coding efficiency.

3. HEVC Intra Coding Process

HEVC introduces a new Coding Tree Unit (CTU) as the main coding structure. A CTU can be as large as 64×64 pixels and can be split into smaller Coding Units (CU) and Prediction Units (PU) through a quadtree structure.

Fig. 1 shows an example of this structure. It can be observed that a CTU can include CUs from 64×64 pixels down to 8×8 pixels. For intra coding, HEVC adopts 35 intra modes, which provide an accurate texture modeling. Many algorithms used for HEVC encoding, such as the ones used in HEVC test model (HM) [42], find the best CTU partitioning through hierarchically processing all possible CU depths (D0 to D3 in Fig. 1) and choosing the best one via rate-distortion optimization. This process is extremely heavy in terms of computations as it includes the predictions for all possible sub-blocks. Since this process forms the major part of the encoding complexity, fast CU partitioning is considered to be the most effective way to reduce or control the coding complexity [31].

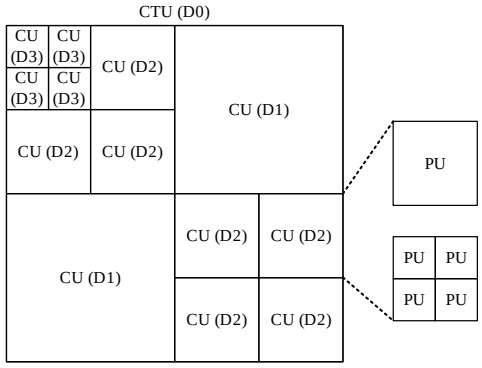

Fig. 1 CTU structure and possible intra PUs

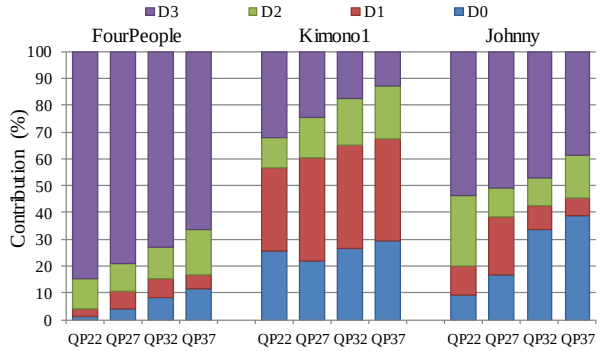

Fig. 2 CU size distribution for FourPeople, Kimono1 and Johnny

To investigate this complexity, the CU size distribution in several video sequences has been analyzed in this research. Fig. 2 summarizes this investigation for three video sequences with four Quantization Parameters (QPs). For Kimono1 which includes mainly homogenous low energy texture, larger CU sizes have a considerable contribution. FourPeople on the other hand includes a rather detailed scene and sharp texture and is mainly coded with smaller CUs. It can be observed that the distribution of CU size varies very much based on the video content. In short, it is observed that blocks with sharp edges, high energy textures, and non-homogeneous content, are more

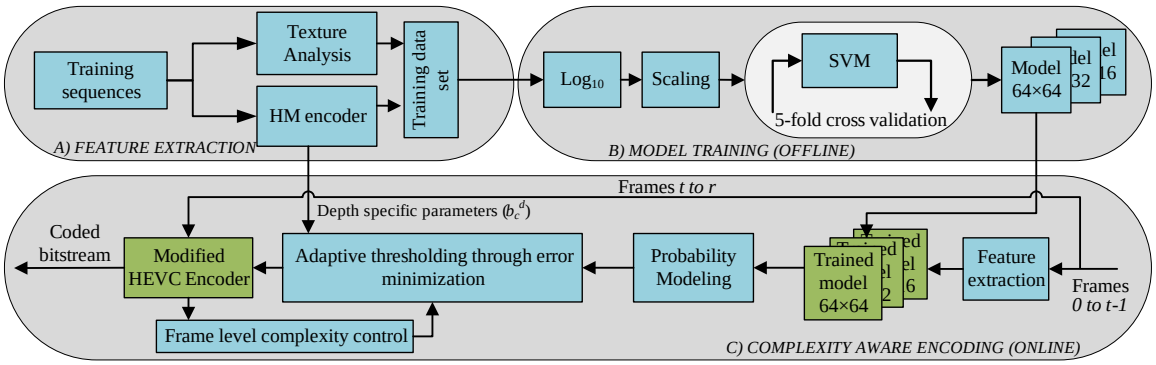

Fig. 3 Overall framework of the proposed method

probable to be coded with smaller CUs.

It can also be observed that the contribution of larger CUs increases with the increase of QP. This is expected, since higher precision is employed with lower QP values and thus small details can lead to a further CU split.

To reduce the complexity in a CU partitioning process, two main decisions can be made, CU Skipping (CUS) [3], and CU partitioning Termination (CUT) [8]. In CUS, the current depth is found to be unsuitable and thus the current CU size is skipped (not processed), while the algorithm continues processing the sub-CUs. On the contrary, in CUT, the current CU is found to be the smallest CU size that might be suitable, and thus the process will be terminated with no further splitting.

To measure the potentials of CUS and CUT for complexity reduction, these techniques have been performed on different CU depths of the same three video sequences of Fig. 2, and the results are summarized in Table 1, where b_0^d and b_1^d denote time saving (equivalent to complexity reduction throughout this paper) of CUT and CUS in depth *d*, respectively. It is observed that the best complexity reduction can be achieved through the termination (CUT) in lower CU depths. Specifically, CUT at depth 0 (D0) leads to 85.23% and at D1 to 73.58% complexity reduction on average. This is expected, as termination in the current depth means that the process of all smaller CUs is avoided. For Skipping (CUS), an additional gain can be achieved for higher depths, due to the larger number of CUs. CUS leads to a smaller complexity reduction (specifically 10% and 13% for D1 and D2), since it only avoids the process of the current CU size. Furthermore, as the table suggests, the complexity reduction of these two techniques is almost the same for different video sequences.

Table 1
Time saving (%) obtained via CUT and CUS at different depths

Sequence	CUT at D0	CUS at D0	CUT at D1	CUS at D1	CUT at D2	CUS at D2
	b_0^0	b_1^0	b_0^1	b_1^1	b_0^2	b_1^2
FourPeople	-86.5	-8.19	-74.41	-10.56	-61.17	-12.99
Johnny	-84.46	-10.03	-72.52	-10.87	-58.86	-13.22
Kimono1	-84.74	-9.38	-73.8	-11.45	-61.06	-13.92
AVG	-85.23	-9.2	-73.58	-10.96	-60.36	-13.38

Some important conclusions can be drawn from the observations in Fig. 2 and Table 1: 1) The encoding complexity can be effectively reduced through CUT and CUS. 2) HEVC encoders (including HM) usually repeat similar coding operations for all CTUs. Hence, the coding complexity reduction due to CUT and CUS does not significantly depend on the video content, and the target coding complexity can be modeled based on a combination of these two decisions. 3) In contrast to the coding complexity, the optimum CU size highly depends on texture characteristics and compression ratio, e.g. edge complexity, texture energy, texture homogeneity, and QP. Thus, the complexity control scheme should take all these features into account in order to achieve the best coding efficiency.

With these observations and motivations, a novel complexity control scheme is proposed in the next section, where a machine learning method is employed to exploit the texture characteristics in order to deliver the best coding efficiency within each target complexity.

4. Proposed Method

The complexity control in this paper is done through an adjustable CU partitioning process. The overall framework of this method is depicted in Fig. 3. First, a feature set is designed to measure various aspects of the texture complexity. These features are used to train SVMs to find the probability of splitting or terminating CUs at each depth level. Next, the distributions of these probabilities are modeled to ascertain the relationship between the coding complexity and coding efficiency. The encoding complexity is also modeled based on the percentages of skipped or terminated CUs in each depth. Finally, the complexity control is modeled as a constrained optimization problem, where the loss of coding efficiency is minimized, according to the obtained distribution, while constraining the complexity to meet the target complexity.

Throughout the paper the term “computational complexity”, or simply “complexity”, is equivalent to and sometime for the ease of reading is exchanged with the “encoding time”. Whereas, the term “texture complexity” refers to the level of details in the picture and should not be confused with the computational complexity. The following sub-sections detail the proposed method.

Table 2 summarizes the important notations used throughout the paper. Scalars are denoted in plain fonts, while bold lowercases show vectors, and bold uppercases show sets of scalars.

4.1 Feature design for texture complexity

Appropriate features are required to perform classification for the CU partitioning process. As a complicated context-

Table 2
Summary of notations used in the paper

Notation	Definition
$\psi(.)$	Dual-tree complex wavelet transform (DT-CWT)
$\phi(.)$	Low-pass filter
ψ_s	Sub-band *s* of *X*, derived from applying DT-CWT to *X*
$\psi_s(i,j)$	Element (*i*,*j*) of sub-band *s*
X	A block of luma samples
X^b	b^{th} sub-block of *X*
$\overline{\psi}_{col\ j}$	Mean of elements in column *j* of ψ
$\boldsymbol{E}$	Set of five features, E_i
$\boldsymbol{w}$	Vector of weights for SVM
ξ	Slack variable for SVM
$\boldsymbol{x}_i$	i^{th} features vector
y_i	i^{th} sample label
b	Bias for SVM
$\hat{\boldsymbol{w}}, \hat{b}$	Trained SVM weights and bias
$\boldsymbol{R}$	Set of six resolution dependent factors, R_c^d, for different classes and depths
$s_{c,i}^d$	Classification score of class *c* and depth *d*, for sample *i*
S_c^d	Threshold of classification for class *c* and depth *d*
$\boldsymbol{S}$	Set of six thresholds, S_c^d, for different classes and depths
t	Number of training frames
r-t	Number of non-training frames after *t* training frames
$\varphi(.)$	Gaussian function
$F_c^d(S)$	CDF of classification scores for class *c* and depth *d*, using threshold *S*
b_c^d	Average complexity reduction for class *c* in depth *d* (as given in Table 1)
$CR(\boldsymbol{S})$	Encoding complexity ratio using set of thresholds $\boldsymbol{S}$
$CRR(\boldsymbol{S})$	Encoding complexity reduction ratio using set of thresholds $\boldsymbol{S}$
CR_f	Measured encoding complexity ratio for frame *f*
CR_T	Target complexity ratio
CR_{bias}	Bias value to adjust the target complexity, to compensate the complexity control error
CR_a	Predefined tolerable complexity control error

dependent phenomenon, CU partitioning can hardly be modeled by a single feature. However, according to the observations in section 3, CUs with high edge complexity, high texture energy, non-homogenous content, and low compression ratio, are more probable to split into sub-CUs. Hence, these criteria were used to design a feature set, which measures different aspects leading into CU partitioning.

The feature set is designed based on filters of a Dual-Tree Complex Wavelet Transform (DT-CWT). In our previous work [17], we demonstrated the effectiveness of DT-CWT analysis for intra mode decision and showed that it results from its directional selectivity, near shift invariance, and less oscillation around singularities [19]. Here, DT-CWT is used to design new features for a different task than [17], which is fast CU partitioning.

Assuming $\psi_h(x)$ and $\psi_g(x)$ to be two real wavelets, with, *h(n)* and *g(n)* as a pair of biorthogonal filter sets, the DT-CWT is defined as (1):

$$\psi(x) = \psi_h(x) + j\,\psi_g(x) \tag{1}$$

Four directional sub-bands through (2) and (3) are defined, where $\phi(.)$ is a low pass filter, $\overline{\psi(.)}$ represents the complex conjugate of $\psi(.)$, and |.| denotes the magnitude. A low frequency band is also defined in (4). DT-CWT applies a subsampling (decimation) similar to most real discrete wavelet transforms. Hence, (2)-(4) define sub-bands with half samples in each dimension, compared to the input signal.

$$\psi_1(x,y) = |\phi(x)\psi(y)|,\ \psi_2(x,y) = |\phi(x)\overline{\psi(y)}| \tag{2}$$

$$\psi_3(x,y) = |\psi(x)\phi(y)|,\ \psi_4(x,y) = \left|\psi(x)\overline{\phi(y)}\right| \tag{3}$$

$$\psi_0(x,y) = \phi(x)\phi(y) \tag{4}$$

Due to high-pass filtering in the vertical direction, ψ_1 and ψ_2 represent two different *nearly* vertical frequency bands (equivalently horizontal bands in the pixel domain), and the opposite is for ψ_3 and ψ_4. More in-depth information on DT-CWT can be found in [19] and [17].

Assuming $\psi_s(i,j)$ represents the element in row *i* and column *j* of sub-band *s*, which is derived for a block of luma, *X*, the set of features $\boldsymbol{E}=\{E_1, E_2, E_3, E_4, E_5\}$ is defined for CU partitioning. Features E_1 and E_2 in the following paragraphs measure the energy-related complexity, E_3 and E_4 measure the homogeneity-related metrics, and E_5 measures the quantization which affects the compression ratio.

1) E_1: Edge complexity

E_1 is defined as the sum of edge energy in four directional sub-bands, as in (5). $DVAR_V$ and $DVAR_H$ are the directional variances in vertical and horizontal directions, as defined in (6). $\overline{\psi}_{col\ j}$ ($\overline{\psi}_{row\ j}$) represents the mean of the values in column (row) *j* of ψ (a sub-band of *X* after DT-CWT), and *n* and *m* are the number of elements of ψ in each dimension.

$$E_1(X) = DVAR_V(\psi_1) + DVAR_V(\psi_2) + DVAR_H(\psi_3) + DVAR_H(\psi_4) \tag{5}$$

$$DVAR_V(\psi) = \frac{1}{mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left(\psi(i,j) - \overline{\psi}_{col\ j}\right)^2$$
$$DVAR_H(\psi) = \frac{1}{mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left(\psi(i,j) - \overline{\psi}_{row\ i}\right)^2 \tag{6}$$

The intuition behind this is that, ideally, if a CU contains pure vertical edges, it would be reflected in the vertical bands (equivalent to horizontal frequency bands) as large elements and thus a common variance will show a large energy. However, the HEVC encoder is able to model such texture with a vertical intra prediction, which ideally leads to a perfect prediction and thus no splitting is required. Hence, E_1 suggests that in order to follow the HEVC behavior, for vertical bands only the variance in the vertical direction should be considered and similarly, for horizontal bands only the variance in horizontal direction should be considered.

2) E_2: Texture complexity

To measure the texture complexity, the effect of sharp edges is excluded through the analysis of the low frequency band (ψ_0), as (7). Variance of ψ_0 represents the global energy of CU.

$$E_2(X) = VAR(\psi_0) \tag{7}$$

3) E_3: Texture homogeneity

Non-uniform texture in a CU can lead to a splitting decision. Thus, the range of texture energy in sub-blocks of the parent

block is measured through (8), where X^b denotes the bth sub-block of X.

$$E_3(X) = \max_{b=1:4}(E_1(X^b)) - \min_{b=1:4}(E_1(X^b)) \quad (8)$$

4) E_4: CU size efficiency

To measure the efficiency of the parent CU compared to the sub-CUs, (9) subtracts the sum of the energies in the four sub-CUs from the energy of the parent CU. In case of a uniform texture, the energy of the parent CU is almost equal to the sum of the sub-CU energies. Whereas in complex textures, these two energies tend to be different.

$$E_4(X) = E_2(X) - \sum_{b=1}^{4} E_2(X^b) \quad (9)$$

5) E_5: Compression ratio

As explained earlier through Fig. 1, the compression ratio has an important effect on CU partitioning. To cover this aspect, the quantizer step size as the main parameter for adjusting the compression ratio is used, as suggested by [3]:

$$E_5 = Q_{step} = 2^{\frac{QP-4}{6}} \quad (10)$$

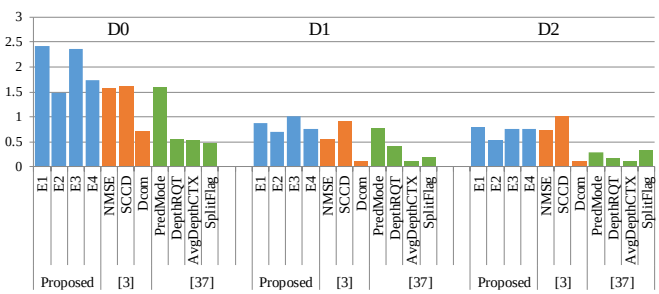

Fig. 4 Average Fisher-score for the training data set, in CU depth 0, 1 and 2

To ensure the feasibility of these features for SVM classification, the Fisher score is used which has been reported to be an efficient measure for SVM classification [37][43]. The Fisher score measures the discrimination of a data set over binary classification, using a feature. Fig. 4 compares the proposed feature set with the ones used in some of the recent CU partitioning schemes [3][37]. In this figure, NMSE, DCom and SCCD are the features representing texture complexity used in [3]. PredMode, $Depth_{RQT}$, $AvgDepth_{CTX}$, and SplitFlag are encoder level information, from the current and neighboring blocks, used in [37]. It can be observed that the proposed features achieve the highest Fisher scores, meaning they are more discriminative. This indicates that they can be effectively used for SVM classification, for CU partitioning.

4.2 Adjustable CU classification using SVM

In this paper, SVM classifier is adopted to model CU partitioning. For a CU at depth d, the ideal case is to be classified into one of the two classes: class0 for cases that need to be treated as a whole block (non-split), and class1for blocks that should be split into sub-CUs. However, like many other real-world applications, CU partitioning data is not ideally separable [3][37]. This means that no decision boundary can be found to classify all data instances into their correct classes. For this reason, two main SVM approaches have been adopted by researchers. The first approach, depicted in Fig. 5 (a), is to classify data with a single SVM line. A variation of this approach is used in [37] that shifts the decision line towards one class, to favor CU splitting. The second approach, depicted in Fig. 5 (b), is to use class specific optimization to separate the noisy data region [3]. Doing so, the data is classified into three classes: class0 for non-split (CUT), class1 for split (CUS), and class2 for undecided. While the encoding of the first two classes is accelerated, the baseline coding is adopted for the undecided class. Moving the two hyperplanes through parameter setting, various trade-offs can be achieved between complexity reduction and coding efficiency. Although the latter approach can provide flexible decision boundaries, this decision is made at the time of training, and hence cannot provide flexibility at the time of encoding, which is required for controlling the complexity.

To provide this flexibility during the encoding, a third approach is proposed here, as depicted in Fig. 5 (c). In this approach, a single SVM is trained for the best possible classification, similar to the first approach. Then, during encoding, instead of a simple binary classification, the distance of each sample from the hyperplane is measured, and an adaptive thresholding is applied on it for classification. The dashed lines in Fig. 5 (c) represent two hypothetical thresholds for class0 and class1. This way a CU is classified into three classes of split, non-split, and undecided. A large distance between a sample and the hyperplane, indicates that the sample can be classified into that class with high confidence. Based on this, threshold lines farther from the hyperplane provide more accuracy, and also less complexity reduction, since more samples will fall into the undecided class. The important advantage of this approach is that the thresholds for classification can be adjusted during encoding, according to the available processing power and the video content, which is useful for complexity control.

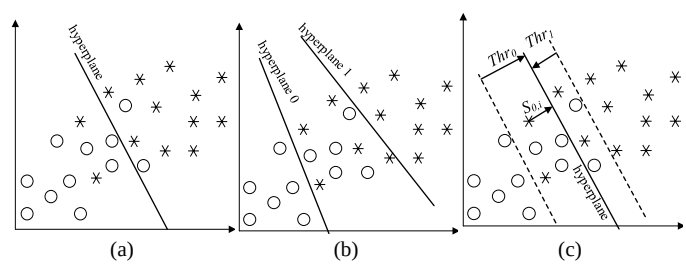

Fig. 5 SVM classification approaches for CU partitioning a) single SVM b) dual-SVM c) proposed adjustable SVM classification

To train the SVM, first, the training data was collected by calculating the feature set $\boldsymbol{E}$, for four different video sequences. Then the training data was labeled with partitioning decisions (ground truth) collected through encoding of these sequences, using the HM encoder, with QP values of 22, 27, 32, and 37. Before feeding these data into the SVM, Log_{10} is applied to all features as suggested in [3], which normalizes the data for linear classification, and then all features are scaled into the same range of [-1,1].

To find the best SVM hyperplane, defined with weights, $\boldsymbol{w}$, and bias, b, for each depth level the following optimization problem in (11) is solved. In this equation, $\boldsymbol{x_i} \in \mathbb{R}^5$, $i=1,\ldots,l$, are training vectors and $y \in \mathbb{R}^l$, are sample labels. ξ_i are slack

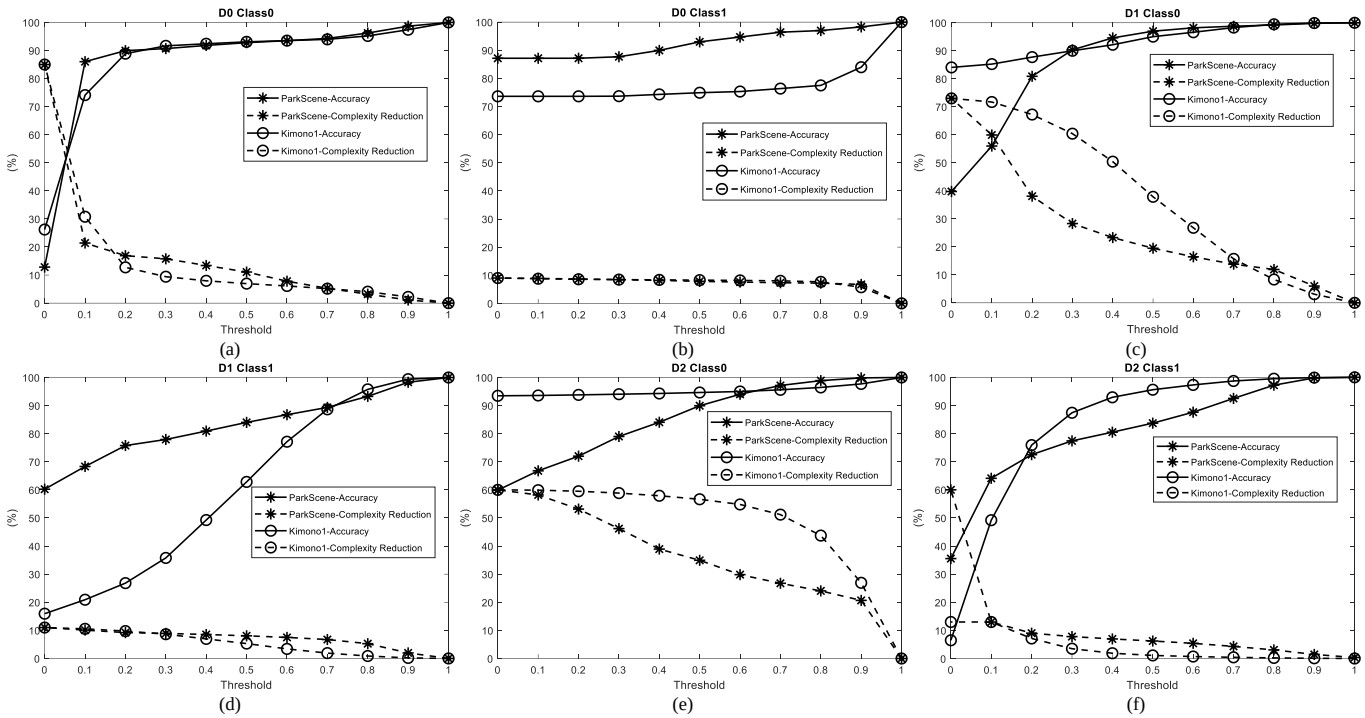

Fig. 6 Classification accuracy and time saving of Kimono1 and ParkScene, for different threshold values of (a) depth0 class0 (b) depth0 class1 (c) depth1 class0 (d) depth1 class1 (e) depth2 class0 (f) depth2 class1

variables for penalizing misclassifications, and C controls the weights of the two terms of the optimization.

Solving this optimization, which is explained in details in [44], obtains the best hyperplane which is denoted with the weights, $\hat{\boldsymbol{w}}$ and bias, $\hat{b}$. Furthermore, to ensure learning generalization, a 5-fold cross validation is performed.

$$\begin{cases} min\,\{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i\} \\ s.t.\ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i,\ \ \xi_i \geq 0, i = 1, \dots, l \end{cases} \tag{11}$$

After the training phase, for evaluating each testing instance with a feature set of $\boldsymbol{x}_i$, first the distance to the hyperplane is measured using (12). Then this score is used to find the posterior probability for each class, c, using (13) [45]. A and B in (13) represent the slope and the intercept respectively, see [45] for more details. As choosing smaller or larger CU sizes lead to different losses of coding efficiency in different resolutions, $\boldsymbol{R} = \{R_0^0,\ R_1^0,\ R_0^1,\ R_1^1,\ R_0^2,\ R_1^2\}$ is used as a resolution dependent factor to balance the probabilities towards the CU sizes that are more suitable for each resolution, for class c at depth d. The following sets of values have been found experimentally for each resolution class: Class A: {1, 0.8, 1, 0.9, 1, 0.5}, class B, C and E: {1, 1, 1, 1, 1, 1}, and class D {0.1, 1, 0.3, 1, 0.7, 1}. Furthermore, due to the binary classification, it is obvious that the probability of class0 and class1 are complementary. Values near 1 for $s_{c,i}$ indicate that sample i is far from the hyperplane and thus can be safely classified into class c, and in the opposite class for values near zero. Whereas, values in between (e.g. 0.5) indicate that the sample is near the decision boundary and a classification can still be performed, but with a lower confidence level.

$$\hat{y}_i = \hat{\boldsymbol{w}}^T\boldsymbol{x}_i + \hat{b} \tag{12}$$

$$s_{c,i} = R_c^d \times P(\hat{y}_i = c|x_i) = R_c^d \times \frac{1}{1+\exp\,(A\hat{y}_i+B)}\ , c = \{0,1\},$$
$$s_{0,i} + s_{1,i} = 1 \tag{13}$$

Finally, for each CU depth d, CU partitioning is decided through (14). If the probability of a sample i for class0, $s_{0,i}^d$, is greater than a threshold, S_0^d, CU is classified into class0 (associated with CUT). If this probability for class1, $s_{1,i}^d$, is greater than a threshold, S_1^d, it is classified into class1 (associated with CUS); otherwise, it is put in class2 (meaning undecided, leading to baseline encoding process). By properly setting the values of S_0^d and S_1^d, one can reach the desired balance between classification accuracy and complexity reduction.

$$\begin{cases} Class0\text{: } Terminate\ (CUT), & if\ s_{0,i}{}^d \geq S_0^d \\ Class1\text{: } Split\ (CUS), & if\ s_{1,i}{}^d \geq S_1^d \\ Class2\text{: } Undecided\ (baseline\ coding), & otherwise \end{cases} \tag{14}$$

Fig. 6 shows the classification performance and complexity reduction, for two video sequences, using different values of S_0^d and S_1^d. Each plot shows the performance for different thresholds on a specific depth and class, while other depths and other classes are processed with the baseline coding method. As expected, increasing the thresholds improves the accuracy and reduces the complexity reduction. However, this effect depends on the video content. For instance, it is observed that for class0 in almost all depths, the performance of ParkScene drops faster than that of Kimono1, whereas, the reverse is true for class1. This is explained by the rather detailed texture of ParkScene compared to the plain texture of Kimono1. For plain textures,

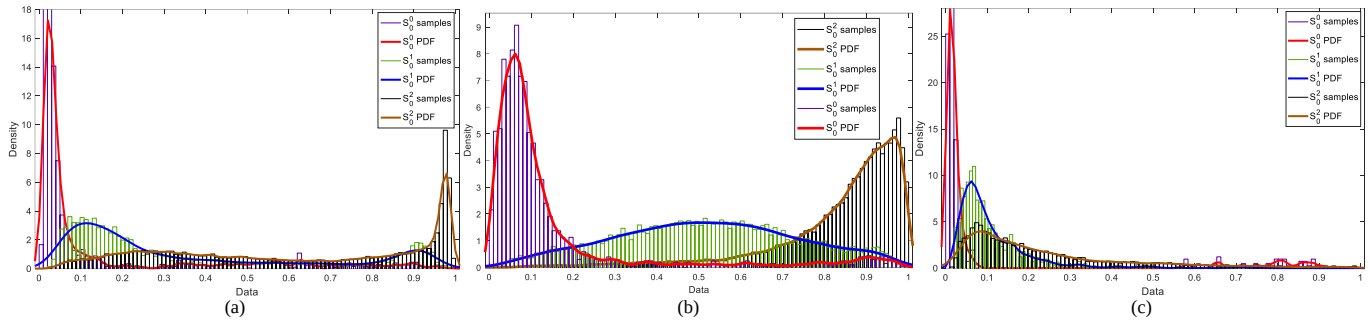

Fig. 7 Distribution of SVM score for class0, in (a) ParkScene, (b) Kimono1, and (c) PartyScene

such as Kimono1, larger CUs are more suitable, thus CUT associated with class0 has a better performance. A similar argument holds for CUS (class1) and detailed textures. Choosing the right threshold value depends on the desired target complexity as well as the video content. In the next section, a method is presented to choose the suitable threshold values adaptively, based on the video texture content.

4.3 Complexity control through adaptive thresholding

Based on the proposed adjustable classification method in section 4.2, the encoding complexity (time) can be controlled, by finding the right threshold values. While different combinations of these values can lead to the same target complexity, as suggested by Fig. 6, these values can differently affect the coding quality of different video content. To maintain quality while controlling the complexity, the distribution of the classification score (probability), i.e. distribution of s_c^d, for t training frames ($t=1$ in our experiments) is modeled, and used to adaptively find the best thresholds for the next r-t frames ($r=100$ in experiments).

The bars in Fig. 7 show the distribution of classification scores s_0^d (class0) for the first frame of ParkScene, Kimono1, and PartyScene. As expected, this distribution is different for different depths and video contents. For instance, for PartyScene, at all three depths, most instances have probabilities near zero, which suggests that their processes should not be terminated, but the current CU depth can be skipped. This indicates a very detailed scene which should mainly be coded with the smallest CU sizes. For Kimono1 on the other hand, parts of depth0 and 1, and the majority of depth2 have probabilities near 1 and can thus be terminated.

Due to the varying type of these distributions for different videos and different coding depths, a single distribution cannot be used to effectively model all contents. Hence, a Gaussian kernel ($\varphi(.)$) distribution, which is a non-parametric distribution, is used to model them, as formulated in (15) [46]. Here, f is the Probability Density Function (PDF), n is the number of data samples, σ is a bandwidth that sets the width of each Gaussian component, which is obtained based on the data range, and $s_{c,i}^d$ is the ith probability sample collected for class c in depth d. Consequently, the Cumulative Distribution Function (CDF) associated with this PDF can be obtained by (16).

$$f(s_c^d) = \frac{1}{n\sigma}\sum_{i=1}^{n}\varphi\left(\frac{s_c{}^d - s_{c,i}{}^d}{\sigma}\right), \sigma = \left(\frac{max_{i=1}^{n}(s_{c,i}{}^d) - min_{i=1}^{n}(s_{c,i}{}^d)}{n}\right) \quad (15)$$

$$F_c^d(S) = P(s_c^d \le S) = \int_{-\infty}^{S} f(s_c^d)ds_c^d \quad (16)$$

Having F_c^d as the CDF of class c at depth d, and also having b_c^d as the contribution of class c to the complexity reduction, as given in Table 1, the encoding complexity ratio for a given set of thresholds, $\boldsymbol{S} = \{S_0^0, S_1^0, S_0^1, S_1^1, S_0^2, S_1^2\}$, can be formulated as (17) and (18). Here, $CR(\boldsymbol{S})$ is the encoding complexity ratio using $\boldsymbol{S}$ as the set of thresholds. Considering that at each CU depth, complexity reduction is equal to the percentage of blocks being split or terminated multiplied by their associated complexity reduction, the total complexity reduction ratio, $CRR(\boldsymbol{S})$, is the sum of the complexity reductions from CUT and CUS obtained from different CU depths.

$$CR(\boldsymbol{S}) = \frac{Constrained\ Enc\ Complexity}{Unconstrained\ Complexity} = 1 - CRR(\boldsymbol{S}) \quad (17)$$

$$CRR(\boldsymbol{S}) = b_0^0\left(1 - F_0^0(S_0^0)\right) + b_1^0\left(1 - F_1^0(S_1^0)\right) + F_0^0(S_0^0)\left(b_0^1\left(1 - F_0^1(S_0^1)\right) + b_1^1\left(1 - F_1^1(S_1^1)\right)\right) + F_0^0(S_0^0)\,F_0^1(S_0^1)\left(b_0^2\left(1 - F_0^2(S_0^2)\right) + b_1^2\left(1 - F_1^2(S_1^2)\right)\right) \quad (18)$$

To provide the best (possible) RD performance, the thresholds need to be ideally close to one. In other words, ideally CUT or CUS should be performed when the SVM classification predicts the highest probability for them. To this end, (19) defines $Err(\boldsymbol{S})$ as the sum of the squared differences of the threshold values from 1.

$$Err(\boldsymbol{S}) = \sum_{c=0}^{1}\sum_{d=0}^{2}(1 - S_c^d)^2 \quad (19)$$

To obtain the best set of thresholds, $\boldsymbol{S}$, the following optimization problem in (20) is solved, where $Err(\boldsymbol{S})$ is minimized, while the encoding complexity ratio, $CR(\boldsymbol{S})$, is constrained to be equal to the target complexity ratio, CR_T. Since values of S_c^d represent thresholds on a probability, they are bound to be in the range of [0,1]. Moreover, S_0^d and S_1^d are thresholds for the PDFs of two complementary probability distributions (class 0 and 1). Thus, their sum should not be less than 1, to ensure that the samples of the two classes do not overlap (not to be confused with the sum of probabilities for two classes, which is less than or equal to 1).

$$\begin{cases} min\{Err(\boldsymbol{S})\} \\ \text{s.t. } CR(\boldsymbol{S}) = CR_T,\ 0 \le S_c^d \le 1,\ S_0^d + S_1^d \ge 1, \forall d \end{cases} \quad (20)$$

This optimization is solved with the fast Sequential Quadratic Programming (SQP) algorithm. The details of this algorithm

can be found in [47]. Solving this optimization, the set of thresholds, $\boldsymbol{S}$, is obtained and used for encoding of the next frames.

4.4 Frame level complexity control

Even though the proposed method provides an efficient and accurate complexity decision, due to imperfections in PDF modeling, complexity modeling, optimization approximations, and also real time variations, such as slight texture difference between consecutive frames, and variations in the processing power, the final encoding complexity might be slightly different from the target complexity. This difference was measured to be up to 3% and 4% for Kimono1 and PartyScene, respectively. To compensate for this, a complexity control at the frame level is often employed [8]. Section 5.3 and Fig. 10 discuss how the frame-level complexity control strategy proposed here, reduces the above-mentioned errors to less than 1%.

The following frame level complexity control strategy is proposed in this paper. The achieved encoding complexity ratio for frame f, CR_f, is measured for each frame, and its distance from the target complexity ratio is calculated as CR_{err}, through (21). To narrow this distance in the next frame, a bias complexity, CR_{bias}, is defined and added to the target ratio. To do so, CR_{err} and its corresponding CR_{bias} from the past few frames are stored in a table. This way CR_{bias} for the current frame can be interpolated from the two nearest CR_{err} values from previous frames.

To provide a practical control mechanism, we allow complexity variations up to a predefined tolerable error, CR_a, and ignore them (CR_a =1% in our experiments). As depicted in (22), only if CR_{err} is larger than CR_a, a bias complexity, CR_{bias}, is calculated and added to the target complexity ratio when finding $\boldsymbol{S}$ for the next frame (via (20)). For the first frame, where no previous history is available, CR_{err} is used as CR_{bias}. For the following frames, first the two closest entries to the current CR_{err} value are located in the table. Then the value of CR_{bias} is calculated from CR_{bias} values associated with these two entries, via linear interpolation.

$$CR_{err} = CR_T - CR_f \tag{21}$$

$$CR_{bias} = \begin{cases} 0, & if\ CR_{err} < CR_a \\ CR_{err}\ , & if\ no\ history\ available \\ interpolate\ (CR_{err1}, CR_{err2}), & otherwise \end{cases} \tag{22}$$

4.5 Overall algorithm

The Algorithm in Fig. 8 sums up the proposed method of complexity control for the HEVC intra coding. At the start of encoding a video sequence, the first frame is passed through all CU levels, and the classification scores ($s_{c,i}^d$) are collected for the whole frame. Based on these samples, the PDF (and thus CDF) models for different classes and all CU depths are updated. Then the threshold values are calculated for the next frames, via (20). This process is repeated every r frames (r=100 in experiments), and at scene changes [31].

For non-training frames, the proposed method repeats through the hierarchical structure of HEVC. For each CTU, at each CU depth, the features are extracted using (5)-(10) and the classification probabilities are obtained via (13). These probabilities are compared with the threshold values to classify the CU into termination, splitting, or undecided, using (14). If terminated, the process of the current CTU ends and the best partitioning is decided from the processed CU sizes. In case of a skip, the current CU size is not processed, and the algorithm continues with the next depth. If undecided, both the current CU size and the next depth level are processed, as in the baseline encoding algorithm.

After encoding each frame, the encoding complexity ratio is compared to the target complexity ratio, and if needed, the target complexity ratio is updated through addition with CR_{bias} via (22). If the target complexity ratio was updated, the thresholds are updated using (20); otherwise they remain unchanged for the next frame.

5. Experimental Results

To evaluate the performance of the proposed method, it was implemented on top of HM test model 16.9 [42]. Video sequences of the common test conditions [48] have been encoded with the proposed method, and the results were compared with those of state-of-the-art methods [4][9]. For the subjective-driven complexity control (TCSVT_SDCC) method [9], their implementation has been modified to support intra coding, and was tested for the all-intra configuration. For the complexity control of intra coding for industrial applications method (TII_CCII) in [4], the reported results of the paper are obtained using exactly the same testing conditions and version of HM as ours, thus the reported results from the paper are used here. Through all experiments, the all-intra configuration has been adopted. Sub-bands of level 2 DT-CWT decomposition have been used for feature extraction, as described in section 4.1, and a value of 1% has been used for CR_a, as in practical applications. In the training phase, four video sequences with various resolutions and texture characteristics, including ParkScene, Johnny, FourPeople, and BQMall, have been used. A separate training set including Kimono1, FourPeople, and

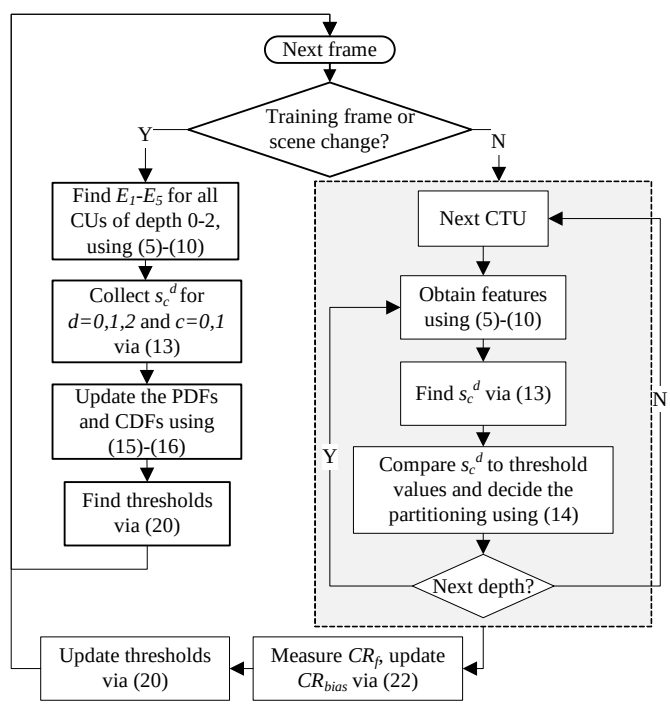

Fig. 8 The final algorithm for complexity control

Table 3
Experimental results with target complexity of 80%

Sequence	Resolution class	TCSVT_SDCC [9]		TII_CCII [4]		Proposed	
		BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)
PeopleOnStreet	A	-	-	0.32	79.6	**0.14**	79.25
Traffic	A	-	-	-	-	**0.09**	80.06
NebutaFestival	A	-	-	-	-	**0.33**	80.86
SteamLocomotiveTrain	A	-	-	-	-	**0.22**	80.18
Kimono1	B	0.05	79.42	**0.03**	80.21	0.04	79.67
ParkScene	B	1.14	81.5	0.29	79.47	**0.08**	80.11
Cactus	B	2.45	80.95	0.26	79.62	**0.05**	79.34
BasketballDrive	B	0.48	81.34	0.29	79	**-0.07**	79.99
BQTerrace	B	1.19	81.05	-	-	**0.07**	80.05
BasketballDrill	C	2.2	80.34	0.68	79.58	**-0.07**	80.29
PartyScene	C	4.84	80.64	0.06	80.16	**-0.02**	80.74
BQMall	C	4.68	78.99	-	-	**-0.01**	80.32
RaceHorses	C	2.45	78.56	-	-	**0.06**	80.27
BQSquare	D	-	-	0.43	79.91	**0.16**	79.23
RaceHorses	D	-	-	0.6	80.42	**0.12**	79.43
BlowingBubbles	D	-	-	-	-	**0.29**	80.14
Johnny	E	0.6	80.48	0.53	80.24	**0.01**	79.86
KristenAndSara	E	1.86	78.55	0.48	79.76	**-0.01**	79.52
Vidyo1	E	1.61	80.3	-	-	**-0.04**	80.34
FourPeople	E	2.68	81.01	-	-	**0.09**	80.3
AVG		2.02	80.24	0.36	79.82	**0.08**	80
AVG_TCSVT		2.02	80.24	-	-	**0.01**	80.06
AVG_TII		-	-	0.36	79.82	**0.04**	79.77

Table 4
Experimental results with target complexity of 60%

Sequence	Resolution class	TCSVT_SDCC [9]		TII_CCII [4]		Proposed	
		BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)
PeopleOnStreet	A	-	-	2.35	60.44	**1.1**	60.25
Traffic	A	-	-	-	-	**0.66**	60.37
NebutaFestival	A	-	-	-	-	**0.43**	59.64
SteamLocomotiveTrain	A	-	-	-	-	**0.2**	60.45
Kimono1	B	0.48	59.52	0.29	60.84	**0.16**	60.31
ParkScene	B	2.66	60.73	1.35	60.52	**0.95**	59.68
Cactus	B	5.24	59.64	2	60.21	**0.75**	60
BasketballDrive	B	3.21	60.57	1.38	59.34	**0.24**	60.69
BQTerrace	B	5.79	58.65	-	-	**0.2**	60.95
BasketballDrill	C	7.51	61.61	2.44	60.24	**0.85**	59.79
PartyScene	C	8.32	58.8	1.83	61.31	**0.58**	60.22
BQMall	C	10.58	59.28	-	-	**0.34**	60.75
RaceHorses	C	6.17	57.75	-	-	**0.74**	61
BQSquare	D	-	-	4.63	60.15	**0.9**	60.46
RaceHorses	D	-	-	3.4	61.19	**3.36**	59.76
BlowingBubbles	D	-	-	-	-	**2.69**	59.23
Johnny	E	3.12	61.22	3.04	61.04	**0.25**	59.68
KristenAndSara	E	3.53	60.27	3.51	60.62	**0.04**	59.92
Vidyo1	E	6.75	60.76	-	-	**0.08**	60.71
FourPeople	E	7.61	57.12	-	-	**0.38**	60.64
AVG		5.46	59.69	2.38	60.54	**0.74**	60.23
AVG_TCSVT		5.46	59.69	-	-	**0.45**	60.28
AVG_TII		-	-	2.38	60.54	**0.83**	60.07

BQMall has been used to evaluate the coding performance of ParkScene and Johnny. Also, ParkScene, Johnny, and Kimono1 were used for training a model to evaluate FourPeople and BQMall, meaning the sequences included in the training set were not evaluated with the same trained model. In the following sections, different aspects of the proposed method are evaluated.

5.1 Encoder performance at constant quality

To evaluate the performance of the proposed method, video sequences were encoded with four target complexities of 80%, 60%, 40% and 20%, and compared to the 100% complexity, i.e. the best HEVC encoding. All experiments were repeated for QP values of 22, 27, 32, and 37, and the Bjotengaard Delta (BD)-Rate and BD-PSNR [49] were measured.

Tables 3 to 6 summarize the coding performance and the measured complexity ratios (CR) for each target complexity. To compare our method with the state-of-the-art methods, these tables also include the results of TII_CCII and TCSVT_SDCC, and provide separate average performance of the proposed method for the sequences reported for each competing method. It can be observed that with smaller target complexities, naturally the BD-Rate increases, since more operations are pruned to meet the target complexity. For the proposed method,

Table 5
Experimental results with target complexity of 40%

Sequence	Resolution class	TCSVT_SDCC [9]		TII_CCII [4]		Proposed	
		BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)
PeopleOnStreet	A	-	-	3.62	39.67	**3.41**	40.38
Traffic	A	-	-	-	-	**3.03**	39.73
NebutaFestival	A	-	-	-	-	**1.41**	39.71
SteamLocomotiveTrain	A	-	-	-	-	**0.65**	39.89
Kimono1	B	1.56	40.84	**0.99**	40.8	1.01	40.45
ParkScene	B	4.69	41.22	2.76	40.12	**2.05**	40.64
Cactus	B	9.24	41.07	4.17	39.91	**2.54**	39.64
BasketballDrive	B	7.33	40.8	4.17	39.91	**2.24**	39.39
BQTerrace	B	10.48	41.64	-	-	**2.36**	39.5
BasketballDrill	C	16.69	38.46	6.38	39.61	**5.83**	40.86
PartyScene	C	11.97	40.93	5.67	41.19	**5.33**	39.69
BQMall	C	16.24	42.29	-	-	**3.59**	40.69
RaceHorses	C	10.29	41.31	-	-	**2.21**	39.36
BQSquare	D	-	-	13.6	40.93	**8.98**	39.47
RaceHorses	D	-	-	10.18	41.9	**5.86**	40.15
BlowingBubbles	D	-	-	-	-	**5.53**	40.56
Johnny	E	12.68	39.5	6.57	40.86	**2.34**	40.76
KristenAndSara	E	14.96	41.27	7.09	40.81	**2.35**	39.65
Vidyo1	E	13.71	41.12	-	-	**2.09**	40.52
FourPeople	E	13.14	39.32	-	-	**1.66**	40.25
AVG		11	40.75	5.93	40.52	**3.22**	40.06
AVG_TCSVT		11	40.75	-	-	**2.77**	40.16
AVG_TII		-	-	5.93	40.52	**3.81**	40.1

Table 6
Experimental results with target complexity of 20%

Sequence	Resolution class	TCSVT_SDCC [9]		TII_CCII [4]		Proposed	
		BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)	BD-Rate (%)	CR (%)
PeopleOnStreet	A	-	-	**9.67**	19.41	13.44	19.94
Traffic	A	-	-	-	-	**8.71**	19.65
NebutaFestival	A	-	-	-	-	**3.61**	20.42
SteamLocomotiveTrain	A	-	-	-	-	**1.32**	20.19
Kimono1	B	2.63	19.55	3.3	19.27	**1.93**	19.24
ParkScene	B	7.89	19.49	6.9	19.83	**6.27**	20.4
Cactus	B	13.87	18.4	13.89	19.96	**13.6**	20.77
BasketballDrive	B	16.01	19.06	13.84	19.56	**11.6**	19.68
BQTerrace	B	15.82	21.03	-	-	**13.69**	19.79
BasketballDrill	C	29.22	19.87	26.29	22.18	**21.38**	19.04
PartyScene	C	19.92	20.53	18.26	22.18	**12.9**	20.01
BQMall	C	22.4	18.81	-	-	**20.46**	19.38
RaceHorses	C	14.09	18.22	-	-	**13.05**	19.96
BQSquare	D	-	-	-	-	**12.95**	20.63
RaceHorses	D	-	-	-	-	**9.23**	20.83
BlowingBubbles	D	-	-	-	-	**7.41**	20.51
Johnny	E	23.89	19.71	16.78	19.56	**14.85**	19.39
KristenAndSara	E	21.29	20.13	**16.35**	19.41	21.55	20.13
Vidyo1	E	17.54	19.43	-	-	**17.22**	19.64
FourPeople	E	16.74	19.17	-	-	**9.11**	19.27
AVG		17.02	19.49	13.92	20.15	**11.71**	19.94
AVG_TCSVT		17.02	19.49	-	-	**13.66**	19.74
AVG_TII		-	-	13.92	20.15	**12.7**	20.01

average BD-Rates for target complexities of 80%, 60% 40% and 20% are 0.08%, 0.74%, 3.22%, and 11.71%, respectively. Compared to previous works, our method gains a considerable improvement, especially in high and middle ranges of power quota. For instance, in 60% target complexity, the proposed method gains 1.55% BD-Rate over TII_CCII and 5.01% over TCSVT_SDCC. This superiority is mainly due to the multi-feature representation of complexity and finer grained complexity modeling of the proposed method.

Compared to TII_CCII which uses the prediction accuracy as the only estimation of texture complexity, the proposed method uses five different features that capture different aspects of complexity. Consequently, the proposed method leads into a fairer complexity allocation in different scenes. More specifically, in scenes with both detailed and plain regions, the single feature of TII_CCII causes over-allocation to the detailed regions, which in turn leads to more degradation in other regions. Such results can be observed in the tables for BasketballDrive, Cactus, and Johnny, where the difference of the two methods is higher than the average of sequences. Moreover, the proposed method decides the CU partitioning at all CU depths, which compared to the CTU level decision of

(a) (b) (c)

Fig. 9 Decoded quality and visualized partitioning decision for parts of BasketballDrive (a) 100% (original HM) (b) 60% and (c) 20% target complexities

TII_CCII (from a predefined set of decisions), provides a wider decision space and hence more accurate decision making. Similar argument can be made for TCSVT_SDCC, which sets a maximum coding depth based on the visual attention. Consequently, the quality of the salient parts of the frames is preserved; however, the quality of other regions drops and the total bitrate also increases.

While the proposed method yields better results than the two mentioned works in most cases, the difference is smaller for smaller target complexities. Ultimately for 20% complexity, the proposed method gains a 1.22% BD-Rate over TII_CCII and 3.36% over TCSVT_SDCC, which compared to the two-digit BD-Rates in 20% complexity, is relatively small. The reason for this is that when only 20% of the processing power is available, most of the baseline operations are pruned, and all three methods need to sacrifice the coding efficiency to guarantee the target complexity; hence, coding options are limited. Moreover, in the case of 80% complexity, small negative BD-Rates are observed for few sequences which might be due to the slight change of entropy or estimation error of BD-Rate calculation and is insignificant to make any conclusions.

The tables also report the achieved complexity ratios for all methods. It can be observed that the proposed method successfully delivers the target complexity ratio, with a very negligible complexity variation. The average control error for the proposed method is 0.00%, 0.23%, 0.06% and 0.06% for target complexities of 80%, 60%, 40% and 20% respectively, which is always below 1%, constrained by the value for CR_a (the tolerable complexity control error, CR_a=1%). The achieved complexity ratio is also the closest to the target complexity among the three competing methods.

5.2 Encoder performance at constant bitrate

To further evaluate the performance of the proposed method for video networking environments, several video sequences were encoded within a target bitrate, under specified target complexity ratios. Table 7 compares the performance of our method, with HM encoder at 100% complexity ratio. It can be observed that the proposed method delivers the target bitrate for all target complexities, while maintaining the target complexity, which justifies the effectiveness of this method. However, to deliver the same bitrate within the limited complexity, the proposed method slightly loses in terms of PSNR. This degradation depends on the complexity level, texture complexity and the target bitrate. Generally, the lower is the target complexity and the higher is the target bitrate, the higher will be the degradation. Also, in detailed scenes, the degradation is higher. For instance, for Kimono1 with a plain texture, the degradation is negligible and in the worst case 0.1 dB (for 20% complexity). For KristenAndSara and BasketballDrill, this degradation is also negligible till 60% complexity. However, in 20% complexity level, the degradation can exceed 1 dB.

Table 7
Performance of proposed method with constant bitrate, compared to HM encoder with 100% complexity

Sequence	Target BR (Kbps)	Target Complexity (%)	Actual BR (Kbps)	PSNR-Y (dB)	Actual complexity (%)	ΔPSNR (dB)
Kimono1	10000	100	9999.84	38.43	100	0
		80	10000.2	38.42	80.26	0
		60	10000.24	38.42	60.83	0
		40	9999.36	38.41	40.83	-0.02
		20	10000.36	38.3	20.54	-0.13
	50000	100	50007.32	43.53	100	0
		80	50009	43.52	80	-0.01
		60	50008.8	43.51	60.92	-0.02
		40	50010.8	43.42	39.71	-0.11
		20	50010.48	43.48	20.52	-0.05
Cactus	10000	100	10006.28	31.1	100	0
		80	10005.84	31.1	80	0
		60	10004.16	31.08	59.63	-0.02
		40	10005.76	31.02	39.11	-0.07
		20	10006.92	30.51	19.18	-0.58
	50000	100	50005.84	37.64	100	0
		80	50002.12	37.64	79.02	0
		60	50001.4	37.6	59.16	-0.04
		40	50003.92	37.51	39.72	-0.13
		20	50010.76	37.04	20.36	-0.6
Krist&Sara	1000	100	1000.92	29.21	100	0
		80	1000.92	29.21	80.12	0
		60	1000.92	29.21	60.7	0
		40	1000.4	29.16	39.5	-0.04
		20	1001.2	28.02	20.44	-1.19

	5000	100	5004.28	37.84	100	0
		80	5003.2	37.83	79.03	-0.01
		60	5003.24	37.76	59.24	-0.09
		40	5003.68	37.69	40.42	-0.15
		20	5006.4	35.67	20.95	-2.17
BasketDrill	1000	100	998.2	28.12	100	0
		80	1000.96	28.14	79.77	0.02
		60	1001.4	28.14	59.87	0.01
		40	999.52	28.06	40.44	-0.06
		20	1000.56	27.3	20.3	-0.82
	5000	100	5008.04	34.79	100	0
		80	4989.92	34.77	79.84	-0.02
		60	5008.84	34.74	59.3	-0.05
		40	5004.04	34.26	40.82	-0.53
		20	5005.6	33.62	19.69	-1.17

This loss of quality is expected, since with a limited processing power, some coding options are pruned, and since the bitrate is targeted to remain intact, the loss is steered to the PSNR. However, even at 20% complexity, this loss can still be reasonably tolerated. Fig. 9 shows parts of the decoded BasketballDrive sequence and the final partitioning decision, when it is encoded at 100%, 60% and 20% complexity ratios. It is observed that even though some degradations can be located for the 20%, through comparison (e.g. degradations around the shoes), the quality is still quite acceptable.

As for the partitioning decision, it is observed through Fig. 9 that the quality at 60% complexity is very similar to that in 100%; however, in smooth areas, some blocks of 64×64 pixels have been skipped to achieve the target complexity. For the 20% though, most of the frames are encoded with blocks of 32×32 and 64×64 pixels. This is because, to save 80% of complexity (nearly the maximum complexity reduction through CU partitioning), most CTUs need to be terminated at depth 0 or 1.

5.3 Frame-level complexity analysis

In this section, the performance of the proposed method is measured through the frames. Fig. 10 shows the encoding complexity ratio for the first 140 frames of two videos. The first 20 frames were coded at 100% complexity, and then every 30 frames at 80%, 60%, 40% and 20% thereafter. The first frame is always used as the training to find the threshold values for the rest of the frames. For this reason, it is always coded with 100% complexity. It is important to note that for the rest of the training frames (e.g. the 100th frame, at scene change, or when the target complexity changes), they do not need to be coded with 100% complexity. While the features and probabilities should be calculated for all sub-blocks, the actual intra prediction process can use the thresholds from previous frames and thus encoding can be done (roughly) with the same timing.

Fig. 10 shows that in all complexity levels, the encoding complexity quickly approaches the target complexity. At the beginning of encoding, or after changing the target complexity ratio, the achieved complexity can be slightly far from the target. This error is less than 3% and 4% for Kimono1 and PartyScene, respectively. However, the frame-level controller compensates this control error through the first few frames after changing the target complexity. Specifically, it is observed that after 2-3 frames, the complexity error falls below 1% (< CR_a), and thus the frame-level controller generally does not change the thresholds and biases through the rest of the sequence.

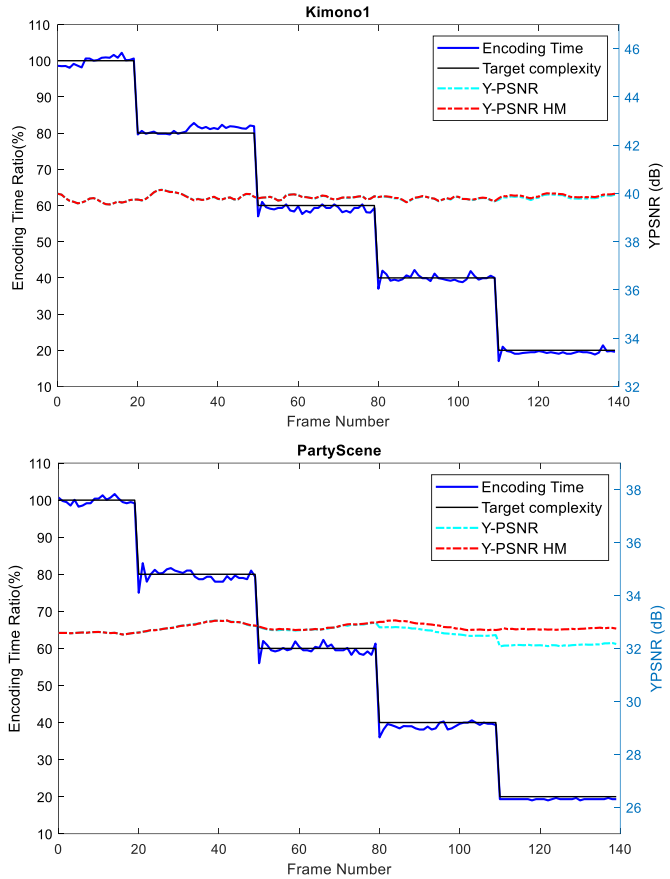

Fig. 10 Performance for first 140 frames of top Kimono1 and bottom PartyScene. Both cases with QP =32

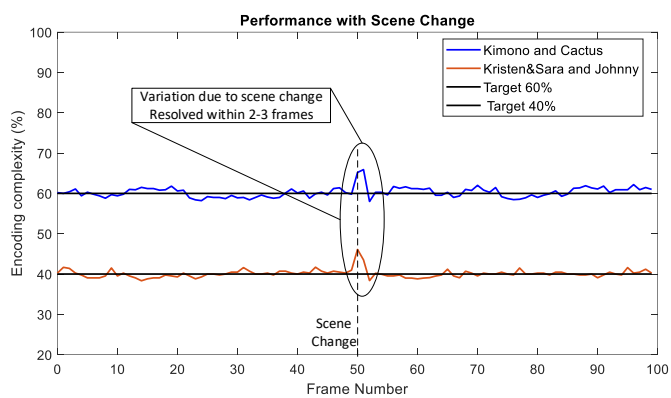

Fig. 11 Performance in presence of scene change

Fig. 10 also compares the quality of constrained coding scenario with HM encoding (100% complexity). A rather uniform distribution of Y-PSNR is observed through the decoded frames, which is important for the perceived quality. For Kimono1 the PSNR remains almost the same throughout all complexity levels. For Kimono1, at 20% complexity, the proposed method loses only 0.02 dB PSNR compared to HM encoding. For the more detailed PartyScene though, a small drop of PSNR is observable when going from 60% to 40% or from 40% to 20% complexity levels (0.22 dB and 0.61 dB respectively). As the figure indicates, PSNR of the complexity control method follows the same trend as HM encoding which shows that the gradual change of PSNR in these figures is related to the gradual change of the video content.

Finally, Fig. 11 shows the coding performance when scene changes occur. Since there are no scene changes in the video sequences suggested by the common test conditions [48], video sequences have been concatenated to create two longer sequence, with a scene change in each. As the figure suggests, the target complexity is met throughout the whole sequence. At the start of each new scene, the controller collects the PDFs for the first frame of the new scene, while this frame is encoded with the thresholds associated with the previous scene. As a result, the target complexity is *approximately* met, even for this frame. Similar to previous tests, the fluctuations at the beginning of the new scene, becomes stable within 2-3 frames.

5.4 Comparison with complexity reduction methods

As explained in sections 1 and 2, the proposed method cannot directly be compared to complexity reduction methods. However, here complexity reduction of a classic SVM method [11], our previous work [17], and a CNN-based method [34] are compared briefly. These methods can gain 25%-58%, 31%-38%, and 57%-66% complexity reductions respectively, while none of them can adjust the complexity during encoding. The proposed method on the other hand adjusts the complexity anywhere between 100% and 20%.

5.5 Analyzing computational overhead

Although the computational overhead of the proposed method has already been included in experimental results, it is measured and discussed separately here, to provide a realistic idea for hardware/software implementation of this method. Similar to all fast decision methods, the proposed method imposes some computational overhead. However, as the HEVC encoding is very computationally demanding, and the proposed feature extraction method is rather simple, the overhead is tolerable. The overhead of the proposed feature extraction method on single thread is ~2% of the overall running time of HM encoding. However, as Wavelet operations can use separable horizontal and vertical filters, they include several parallel 1-D filtering operations which not only have moderate complexity overhead, but they also have a great potential for parallel processing. Using the OpenMP [50] library to parallelize this process on only four processing cores, this overhead is reduced to ~0.5% of the HM encoder on average. Even considering the case of optimized encoding, e.g. 60% and 20% processing power, this overhead would be 0.89%, and 2.67% respectively, which are tolerable. It is important to note that as HEVC encoding can benefit from hardware or software optimizations in commercial products, similar techniques can be applied to the proposed feature extraction unit.

6. Conclusion

In this paper, a novel SVM-based approach is presented to flexibly control the complexity of HEVC intra coding. This approach uses DT-CWT decomposition to design features for CU partitioning. Then an SVM-based approach is proposed to learn the probability of CU partitioning at each CU depth. Finally, an adaptive thresholding technique is used to maximize the coding efficiency within the target complexity. Extensive experimental results confirm the performance of the proposed method within target complexity ratios of 80% to 20%. The achieved encoding complexity in each target complexity was measured to be less than 1% away from the target. Also, through fine-grain decision making at each CU depth, and high-quality features that measure different aspects of texture complexity, the proposed method gains better coding efficiency compared to competing state-of-the-art methods, at all complexity levels.

Acknowledgment

This work has been funded in part by Business Finland, within Project IMD-4072.

References

[1] G.J. Sullivan, J. Ohm, W. Han, T. Wiegand, Overview of the High Efficiency Video Coding, IEEE Trans. Circuits Syst. Video Technol. 22 (2012) 1649–1668.

[2] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Trans. Circuits Syst. Video Technol. 13 (2003) 560–576.

[3] X. Liu, Y. Li, D. Liu, P. Wang, L.T. Yang, An Adaptive CU Size Decision Algorithm for HEVC Intra Prediction based on Complexity Classification using Machine Learning, IEEE Trans. Circuits Syst. Video Technol. 29 (2019) 144–155.

[4] J. Zhang, S.T.W. Kwong, T. Zhao, H.H.S. Ip, Complexity Control in the HEVC Intracoding for Industrial Video Applications, IEEE Trans. Ind. Informatics. 15 (2019) 1437–1449.

[5] G. Correa, P. Assuncao, L. Agostini, L. da Silva Cruz, Complexity control of high efficiency video encoders for power-constrained devices, IEEE Trans. Consum. Electron. 57 (2011) 1866–1874.

[6] M. Abdallah, C. Griwodz, K.-T. Chen, G. Simon, P.-C. Wang, C.-H. Hsu, Delay-Sensitive Video Computing in the Cloud, ACM Trans. Multimed. Comput. Commun. Appl. 14 (2018) 1–29.

[7] S. Schmidt, S. Zadtootaghaj, S. Moller, Towards the delay sensitivity of games: There is more than genres, in: Int. Conf. Qual. Multimed. Exp., IEEE, 2017: pp. 1–6.

[8] A. Jimenez-Moreno, E. Martinez-Enriquez, F. Diaz-de-Maria, Complexity Control Based on a Fast Coding Unit Decision Method in the HEVC Video Coding Standard, IEEE Trans. Multimed. 18 (2016) 563–575.

[9] X. Deng, M. Xu, L. Jiang, X. Sun, Z. Wang, Subjective-Driven Complexity Control Approach for HEVC, IEEE Trans. Circuits Syst. Video Technol. 26 (2016) 91–106.

[10] G. Correa, P.A. Assuncao, L.V. Agostini, L.A. da Silva Cruz, Pareto-Based Method for High Efficiency Video Coding With Limited Encoding Time, IEEE Trans. Circuits Syst. Video Technol. 26 (2016) 1734–1745.

[11] T. Zhang, M.T. Sun, D. Zhao, W. Gao, Fast Intra-Mode and CU Size Decision for HEVC, IEEE Trans. Circuits Syst. Video Technol. 27 (2017) 1714–1726.

[12] R. Rodriguez-Sanchez, M.T. Alonso, J.L. Martinez, R. Mayo, E.S. Quintana-Orti, Time and energy modeling of an intra only HEVC encoder, in: Vis. Commun. Image Process., IEEE, 2015: pp. 1–4.

[13] O. Alaoui-Fdili, Y. Fakhri, P. Corlay, F.-X. Coudoux, D. Aboutajdine, Energy consumption analysis and modelling of a H.264/AVC intra-only based encoder dedicated to WVSNs, in: IEEE Int. Conf. Image Process., IEEE, 2014: pp. 1189–1193.

[14] Apple ProRes, Apple White Pap. (2018) 1–28. https://apple.com/final-cut-pro/docs/Apple_ProRes_White_Paper.pdf.

[15] J. Lei, D. Li, Z. Pan, Z. Sun, S. Kwong, C. Hou, Fast Intra Prediction Based on Content Property Analysis for Low Complexity HEVC-Based Screen Content Coding, IEEE Trans. Broadcast. 63 (2017) 48–58.

[16] L. Xu, S. Kwong, Y. Zhang, D. Zhao, Low-Complexity Encoder Framework for Window-Level Rate Control Optimization, IEEE Trans. Ind. Electron. 60 (2013) 1850–1858.

[17] F. Pakdaman, M.-R. Hashemi, M. Ghanbari, Fast and efficient intra mode decision for HEVC, based on dual-tree complex wavelet, Multimed. Tools Appl. 76 (2017) 9891–9906.

[18] E. Hosseini, F. Pakdaman, M.R. Hashemi, M. Ghanbari, A

computationally scalable fast intra coding scheme for HEVC video encoder, Multimed. Tools Appl. (2018) 1–24.
[19] I.W. Selesnick, R.G. Baraniuk, N.C. Kingsbury, The dual-tree complex wavelet transform, IEEE Signal Process. Mag. 22 (2005) 123–151.
[20] L.L. Wang, W.C. Siu, Novel adaptive algorithm for intra prediction with compromised modes skipping and signaling processes in hevc, IEEE Trans. Circuits Syst. Video Technol. 23 (2013) 1686–1694.
[21] N. Hu, E.H. Yang, Fast Mode Selection for HEVC Intra-Frame Coding with Entropy Coding Refinement Based on a Transparent Composite Model, IEEE Trans. Circuits Syst. Video Technol. 25 (2015) 1521–1532.
[22] J. Gu, M. Tang, J. Wen, Y. Han, Adaptive Intra Candidate Selection With Early Depth Decision for Fast Intra Prediction in HEVC, IEEE Signal Process. Lett. 25 (2018) 159–163.
[23] M. Jamali, S. Coulombe, Fast HEVC Intra Mode Decision Based on RDO Cost Prediction, IEEE Trans. Broadcast. (2018) 1–14.
[24] M.U.K. Khan, M. Shafique, J. Henkel, Fast hierarchical intra angular mode selection for high efficiency video coding, in: IEEE Int. Conf. Image Process., 2014: pp. 3681–3685.
[25] M. Usman, K. Khan, M. Shafique, J. Henkel, An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding, in: Image Process. (ICIP), 2013 20th IEEE Int. Conf., 2013: pp. 1578–1582.
[26] B. Min, R.C.C. Cheung, A fast CU size decision algorithm for the HEVC intra encoder, IEEE Trans. Circuits Syst. Video Technol. 25 (2015) 892–896.
[27] S. Cho, M. Kim, Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding, IEEE Trans. Circuits Syst. Video Technol. 23 (2013) 1555–1564.
[28] K. Lim, J. Lee, S. Kim, S. Lee, Fast PU Skip and Split Termination Algorithm for HEVC Intra Prediction, IEEE Trans. Circuits Syst. Video Technol. 25 (2015) 1335–1346.
[29] Z. Li, Y. Zhao, Z. Dai, K. Rogeany, Y. Cen, Z. Xiao, W. Yang, A fast CU partition method based on CU depth spatial correlation and RD cost characteristics for HEVC intra coding, Signal Process. Image Commun. 75 (2019) 141–146.
[30] W. Liao, Z. Chen, A fast CU partition and mode decision algorithm for HEVC intra coding, Signal Process. Image Commun. 67 (2018) 140–148.
[31] H.S. Kim, R.H. Park, Fast CU partitioning algorithm for HEVC using an online-learning-based Bayesian decision rule, IEEE Trans. Circuits Syst. Video Technol. 26 (2016) 130–138.
[32] L. Shen, Z. Zhang, Z. Liu, Effective CU size decision for HEVC intracoding, IEEE Trans. Image Process. 23 (2014) 4232–4241.
[33] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, D. Wang, CU partition mode decision for HEVC hardwired intra encoder using convolution neural network, IEEE Trans. Image Process. 25 (2016) 5088–5103.
[34] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, Z. Guan, Reducing Complexity of HEVC: A Deep Learning Approach, IEEE Trans. Image Process. 27 (2018) 5044–5059.
[35] D. Lee, J. Jeong, Fast CU size decision algorithm using machine learning for HEVC intra coding, Signal Process. Image Commun. 62 (2018) 33–41.
[36] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, Z. Peng, Binary and multi-class learning based low complexity optimization for HEVC encoding, IEEE Trans. Broadcast. 63 (2017) 547–561.
[37] M. Grellert, B. Zatt, S. Bampi, L.A. da S. Cruz, Fast Coding Unit Partition Decision for HEVC Using Support Vector Machines, IEEE Trans. Circuits Syst. Video Technol. 29 (2019) 1741–1753.
[38] G. Correa, P. Assuncao, L.A. da Silva Cruz, L. Agostini, Adaptive coding tree for complexity control of high efficiency video encoders, in: 2012 Pict. Coding Symp., IEEE, 2012: pp. 425–428.
[39] W. Penny, I. Machado, M. Porto, L. Agostini, B. Zatt, Pareto-based energy control for the HEVC encoder, in: 2016 IEEE Int. Conf. Image Process., IEEE, 2016: pp. 814–818.
[40] X. Deng, M. Xu, C. Li, Hierarchical Complexity Control of HEVC for Live Video Encoding, IEEE Access. 4 (2016) 7014–7027.
[41] J. Zhang, S. Kwong, T. Zhao, Z. Pan, CTU-Level Complexity Control for High Efficiency Video Coding, IEEE Trans. Multimed. 20 (2018) 29–44.
[42] C. Rosewarne, B. Bross, M. Naccari, K. Sharman., G. Sullivan, High efficiency video coding (HEVC) test model 16 (HM16) improved encoder description update 5, JCTVC-W1002. (2016).
[43] Y.-W. Chen, C.-J. Lin, Combining SVMs with Various Feature Selection Strategies, in: Featur. Extr., Springer Berlin Heidelberg, Berlin, Heidelberg, 2006: pp. 315–324.
[44] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 1–27.
[45] T.-F. Wu, C.-J. Lin, R.C. Weng, Probability Estimates for Multi-class Classification by Pairwise Coupling, J. Mach. Learn. Res. 5 (2004) 975–1005.
[46] A.W. Bowman, A. Azzalini, Applied smoothing techniques for data analysis : the kernel approach with S-Plus illustrations, Oxford University Press Inc, 1997.
[47] M.J. Powell, A fast algorithm for nonlinearly constrained optimization calculations, in: Numer. Anal., Springer, Berlin, Heidelberg, 1978: pp. 144–157.
[48] F. Bossen, Common Test Conditions and Software Reference Configurations, JCTVC-H1100. (2012).
[49] G. Bjontegaard, Calculation of Average PSNR Differences Between RD Curves, 13th Video Coding Expert. Gr. Meet. (2001).
[50] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.