Fast Block Size Decision for HEVC Encoders with On-the-Fly Trained Classifiers

Guilherme Correa, Pargles Dall'Oglio, Daniel Palomino, Luciano Agostini
Video Technology Research Group (ViTech)
Graduate Program in Computing (PPGC)
Federal University of Pelotas (UFPel)
Pelotas, Brazil
{gcorrea, pwdalloglio, dpalomino, agostini}@inf.ufpel.edu.br

Abstract— **High Efficiency Video Coding (HEVC) introduced flexible block partitioning structures that increased significantly compression rates in comparison to previous standards. However, such features resulted in a non-negligible increase in computational cost as well. To accelerate this complex partitioning process, this paper proposes a method that halts the usual rate-distortion optimization employed in Coding Unit size decision by a set of decision tree classifiers, which are trained on the fly according to the current video sequence characteristics. The classifiers are built during the encoding process by the C5 machine learning algorithm, which was chosen based on an extensive analysis that compared several algorithms in terms of decision accuracy and training complexity. Experimental results show that the strategy is capable of building accurate models and decreases the HEVC encoding time in 34.4% on average, at the cost of a compression efficiency loss of only 0.2%.**

Keywords— video coding, HEVC, complexity reduction, decision trees, machine learning

I. Introduction

The High Efficiency Video Coding (HEVC) standard [1] was developed by the Joint Collaborative Team on Video Coding (JCT-VC) and launched in 2013 with the aim of duplicating the compression rates achieved by its predecessor, the H.264/AVC standard, while still maintaining the same level of subjective image quality. To reach such goal, several new compression techniques and modes have been included in the standard specification [1] and implemented in the HEVC Test Model (HM) [2], effectively increasing the computational cost of HEVC in comparison with H.264/AVC in up to 500%, depending on the encoder configuration [3].

The employment of more flexible partitioning structures has been claimed as the main responsible for the compression gains achieved by HEVC but has also been charged as the standard's most computationally demanding inclusion in recent works that analyze the encoder complexity [3, 4]. To achieve the best compression efficiency, the HM encoder runs an exhaustive search process which tests every possible combination of partitioning structures and chooses the one that results in the lowest rate-distortion (R-D) cost. Although the most recent HM versions include some speedup features, this process is still responsible for most of the encoder's computational cost.

Several authors have proposed heuristics for simplifying partitioning decisions in HEVC [5-7]. However, most of these heuristic-based approaches result in non-negligible losses in compression efficiency. To minimize such losses, intelligent strategies that apply machine learning techniques to extract and analyze image characteristics and intermediate encoding results have been proposed more recently, especially for transcoding [8, 9] and encoding cost reduction [10-12]. However, these works are all based on an offline training approach that uses information from a set of previously encoded video sequences to build models that are statically implemented in the encoder and then used to encode new video sequences. The offline training-based approach proposed in [12] presents the best results among those found in the literature that employ machine learning techniques for computational cost reduction in HEVC.

Differently from [12] and other related works, this paper proposes an online training scheme that allows fast decision models to be trained and built in the first steps of the HEVC encoding process. The models are specifically trained for each video sequence and periodically retrained to accommodate scene changes, dynamically adjusting themselves to the video sequence characteristics. This work first presents an analysis over a set of different machine learning algorithms that was performed to select the one that yields the most accurate models given a set of features collected from the HEVC encoding process. The C5 algorithm [13] was selected and implemented in the HM encoder, leading to an average computational cost reduction of 34.4% with an average bit rate increase of only 0.2% in comparison to the original HM encoder.

The paper is organized as follows: section II presents a short overview of the HEVC partitioning structures. Section III presents the information gain-based attribute evaluation performed and the analysis over a set of candidate machine learning algorithms for implementation in the HEVC encoder. Section IV presents the proposed video coding flow and details the C5 algorithm implementation in the HEVC encoder. Experimental results are presented in section V and conclusions are discussed in section VI.

II. Frame Partitioning in HEVC

Even though HEVC maintains the hybrid coding architecture used in previous standards, its frame partitioning structures are significantly different. To better cope with image regions with different characteristics, each frame is adaptively divided into blocks of different sizes in a recursive quadtree splitting process.

Each frame is first divided into equal-sized square blocks called Coding Tree Units (CTU), usually of 64×64 pixels, which are used as roots for each coding quadtree (or coding tree). Each leaf of the coding tree is called a Coding Unit (CU) and its dimensions can vary from 8×8 up to the CTU dimensions, depending on the tree depth at which it is located. The smaller the coding tree depth, the larger is the coding unit. Ideally, the best coding tree configuration can be found only through a Rate-Distortion Optimization (RDO) process which evaluates every possible quadtree configuration and compares all of them in terms of bit rate and image quality. Fig. 1(a) and Fig. 1(b) shows an example of a 64×64 CTU divided in a quadtree structure into 13 CUs, with dimensions varying from 32×32 to 8×8 pixels.

CUs are further subdivided into other entities for prediction, called Prediction Units (PU), and for prediction residue transformation, the Transform Units (TUs). Each CU can be predicted separately as one, two or four PUs using inter-frames or intra-frame prediction. Although PUs are not organized in a quadtree structure, the best PU division is also determined through exhaustive iterations of the RDO process taking into consideration the resulting bit rate and image quality for each PU division possibility. Fig. 1(c) shows a 32×32 CU divided into two 16×32 PUs for prediction.

During the transform coding of the prediction residual, each CU is assumed to be root of another quadtree-based structure called residual quadtree (RQT). Each leaf of the RQT is called a Transform Unit (TU), which sizes can vary from 4×4 up to the CU size, depending on the depth at which the TU is located. Similarly to the coding tree, the RQT structure is also defined by exhaustive RDO iterations. Fig. 1(d) shows an example of a 32×32 CU divided into 19 TUs, whose dimensions vary from 16×16 to 4×4 pixels.

The number of coding tree possibilities grows exponentially with the maximum depth allowed. Moreover, for each possible CU in each possible coding tree configuration, all possible PUs and RQT configurations are tested in the RDO process, which involves the computation of bit rate and reconstructed image fidelity after performing intra/inter-prediction, direct and inverse transform and quantization, entropy coding and deblocking operations for each configuration. Limiting the computational burden related to the definition of the encoding structure is thus essential if one aims at optimizing HEVC encoder for operation at or below a given complexity level for use in systems with reduced energy allowances or limited computational power.

III. Attributes and Learning Algorithm Selection

This section presents the data mining methodology employed for selecting the features used in the candidate machine learning models trained for fast encoding decisions. The setup employed in the data mining and training process is also described. As presented in the previous section, most of the complexity associated to the HEVC encoding process is related to the new partitioning structures flexibility, especially the CUs. This way, the analysis and the training process presented in this section aim at discovering which attributes and algorithms yield the best relevance and accuracy regarding the observed CU splitting decision.

The *Waikato Environment for Knowledge Analysis* (WEKA) [14] tool was used in both the data mining and model training processes. WEKA is a free, open-source data mining platform that includes several machine learning algorithms and tools for pre-processing, classifying, clustering, and visualizing the data to be explored. The input for WEKA is an *Attribute-Relation File Format* (ARFF) file, which consists of a header with a list of attribute declarations, followed by a section that contains the raw data with one instance per line and one attribute value per column.

A. Data Mining and Information Gain Evaluation

The first 30 frames of 10 video sequences recommended in the Common Test Conditions (CTC) document defined by the JCT-VC group were used to collect data for the data mining process. The sequences are *BlowingBubbles*, *RaceHorses*, *PartyScene*, *BQMall*, *SlideShow*, *Vidyo1*, *BasketballDrive*, *ParkScene*, *NebutaFestival* and *Traffic*, which differ broadly from one another in terms of temporal and spatial resolution, motion and texture characteristics. The HEVC Test Model (HM) [2] software was used to encode the sequences with QPs 22, 27, 32, and 37, using the Random Access (RA) temporal configuration [15], also defined in the CTC document.

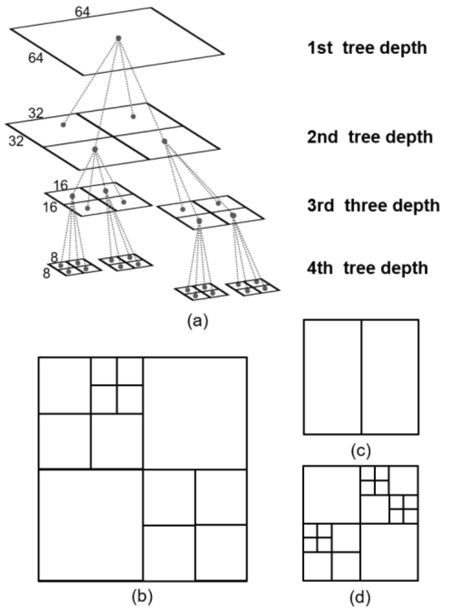

Fig. 1. HEVC partitioning structures: (a) quadtree for a 64×64 CTU, (b) 64×64 CTU divided into 13 CUs, (c) 32×32 CU divided into 2 PUs, (c) 32×32 CU divided into 19 TUs.

Among 40 variables considered in the data mining process, only the 10 most useful were selected for the model training process described in the next subsection. The usefulness of each of these variables for inferring the decision of splitting or not a certain CU was assessed through the Information Gain Attribute Evaluation (IGAE) method in WEKA, which measures the information gain (IG) [16] achievable when using a variable to classify the data into the possible classes. This gain equates to the difference between the number of bits per data item necessary to convey its class identity before and after classification of the data set using decision rules based on the variable in question [16]. Therefore, the IG of a variable indicates how relevant it is for the process of constructing a model that correctly decides to which class each data item belongs. Based on this measure, a manual analysis procedure was followed to identify the most useful variables for the model training. Those variables with higher IG were selected as features for the model training processes, which are explained in the next subsection.

In the specific case of this work, the IG of each variable is computed regarding its contribution to classify correctly each CU as *Split* or *Don't Split*. Table I lists the 10 selected attributes and their respective IG regarding the classification of 64×64, 32×32 and 16×16 CUs as *Split* or *Don't Split*. Attributes *RD(2N×2N)*, *RD(2N×N)* and *RD(N×2N)* correspond to the R-D cost of encoding a CU based on a single square PU (i.e., 2N×2N mode), two vertical rectangular PUs (i.e., 2N×N mode) or two horizontal rectangular PUs (i.e., N×2N mode), respectively. Attribute *RD(MSM)* represents the

R-D cost of encoding a CU under the Merge-Skip Mode (MSM) [1]. Additionally, attributes *Ratio(best, MSM)* and *NormDiffRD(best, MSM)* are the ratio and the normalized ratio between *RD(best)* and *RD(MSM)*, calculated as in (1) and (2), where *RD(best)* is the current best R-D cost among all tested PU modes for the CU.

Attribute *Partition* corresponds to the PU mode selected for the CU, which may assume eight different possible values (2N×2N, 2N×N, N×2N, N×N, 2N×nU, 2N×nD, nL×2N, or nR×2N). This is the attribute with highest IG among all those analyzed, which is expected. in general, if a large PU mode is selected for a given CU, very rarely this CU needs to be split into smaller CUs in the recursive process.

$$Ratio\ (best, MSM) = \frac{RD(Best)}{RD(MSM)} \quad (1)$$

$$NormDiffRD(best, MSM) = \left|\frac{RD(Best) - RD(MSM)}{RD(MSM)}\right| \quad (2)$$

SkipMergeFlag and *MergeFlag* are binary flags that represent the use of the Skip and the Merge modes to encode the current CU. Generally, homogeneous and static areas of a video tend to be encoded as Skip or Merge CUs, mostly as large blocks. Finally, *ΔNeighDepth* is an attribute calculated based on the difference between the current CU depth in the quadtree structure and the maximum depth used in temporal and spatially neighboring CTUs. This is a relevant parameter due to natural spatial and temporal correlation present in video sequences. Usually, small CUs are located in image areas composed of many other small CUs.

B. Machine Learning Algorithm Evaluation

The WEKA toolkit includes more than a hundred machine learning algorithms [14], which are categorized as *Bayes*, *Functions*, *Trees*, *Lazy*, *Rules*, *Meta*, *Multi-instance* and *Miscellaneous*. Several of these categories do not support certain types of data, such as numerical and binary attributes. This way, those algorithms that do not support the type of attributes selected in section III.A were excluded from the evaluation presented here.

As HEVC supports up to four quadtree levels, three different models were trained by each candidate algorithm, i.e. for CUs 64×64, 32×32 and 16×16. To emulate an online training process in this algorithm evaluation phase, three models were trained for each video sequence, taking the information gathered from the encoding of the first 30 frames as training instances. Data collected from the remaining frames of each video sequence (250 frames) were used to validate the trained models, measuring the achieved classification accuracy as the percentage of correct decisions in the total amount of instances used in the test process. The same sequences used in the data mining process presented in the subsection III.A were used in the models training.

More than 50 algorithms were tested in order to find out which of them generate models that achieve the best decision accuracy results with acceptable overhead in the encoder computational cost. Since the main idea of this work consists of implementing the selected training algorithm in the HEVC encoder, the algorithm should be quickly executable; otherwise, computational cost could increase instead of increase. For clarity, Table II lists the average results obtained for the algorithms that belong to the Tree category of WEKA. Although several other algorithms of various categories were also tested, those belonging to the Tree category presented the best results in terms of decision accuracy considering a constrained training time. The average results for each algorithm consider the three models built for each tested sequence, thus totalizing 30 models per algorithm.

TABLE I. INFORMATION GAIN ATTRIBUTE EVALUATION (IGAE) RESULTS.

Attribute	**Information Gain (IG)**		
	64×64	**32×32**	**16×16**
Partition	0.352	0.336	0.269
ΔNeighDepth	0.311	0.262	0.249
Ratio(best, MSM)	0.112	0.168	0.255
NormDiffRD(best, MSM)	0.109	0.163	0.249
RD(2N×2N)	0.035	0.042	0.053
RD(MSM)	0.034	0.061	0.108
RD(2N×N)	0.033	0.036	0.044
RD(N×2N)	0.031	0.032	0.042
SkipMergeFlag	0.046	0.066	0.065
MergeFlag	0.020	0.035	0.046

TABLE II. ACCURACY, FALSE NEGATIVES AND TRAINING TIME FOR ALGORITHMS IN THE TREE CATEGORY.

Algorithm	**Decision Accuracy (%)**	**False Negatives (%)**	**Training Time (s)**
LADTree	85.97	7.23	28.49
ADTree	85.39	8.22	7.00
J48	83.83	9.54	3.51
Decision Stump	80.83	12.71	0.47
REPTree	80.54	14.39	0.60
Rotation Forest	80.42	16.58	44.77
Random Forest	80.36	15.95	57.03

Table II shows that LADTree [14] achieved the best average decision accuracy results (85.97%) but incurs in a very long training time (28.49 seconds) in the WEKA platform. The Random Forest algorithm [14] achieved the lowest decision accuracy (80.36%) and required the longest training time (57.03 seconds). When considering the tradeoff between accuracy and training time, the J48 algorithm [14] is the best option, with a decision accuracy of 83.83% and a training time of 3.51 seconds. For all algorithms in the Tree category, the only parameters configured for the training process were the decision tree limitation to a maximum of 10 decision levels, aiming at constraining the model computational cost, and the 10-fold cross validation used in the training process.

It is important to note that decision accuracy numbers also consider as misclassifications the case of splitting a CU that should not be split into smaller CUs, even though it does not necessarily harm the encoding R-D efficiency. In fact, R-D efficiency is only harmed when a CU that should be split is not split due to the early termination provided by the models. These cases, treated here as false negatives, are presented in the third column of Table II. Note that false negatives are much rarer than the overall inaccurate decisions. Considering the J48 algorithm, the actual errors (false negatives) occur in only 9.54% of the cases.

J48 is the implementation available in WEKA for the C4.5 algorithm used to generate decision trees, which was proposed in 1993 by Ross Quinlan [16]. The C4.5 algorithm starts by taking all instances fed to it as inputs and calculates the information gain of using each attribute to perform the classification using a determined threshold. By iterating among all attributes and adjusting the thresholds, C4.5 measures the information gain of each variable and threshold pair. Then, the attribute (and its corresponding threshold) with the largest information gain is chosen to divide the training data into two sub-sets. The same process is applied recursively to the two sub-sets.

More recently, the authors of C4.5 have proposed C5 [13], which is a more efficient version of the previous algorithm. The authors claim that C5 is able to train shorter (i.e., less complex) decision trees with higher decision accuracy levels. Besides, the C5 algorithm training process is much faster than the observed with C4.5 [13], which makes it a great candidate for online training in video encoders. As C5 is not available in WEKA, a separate analysis comparing C4.5 and C5 using the same input datasets used in the experiment presented in Table II was performed. The C4.5 and C5 algorithm implementations in C language [13] were compared in terms of training time and decision accuracy. The obtained average results, which consider the three trained models per video sequence as previously explained, show that the training with C5 was 54 times faster than with C4.5. Furthermore, average decision accuracy results show that C4.5 achieved an accuracy of 81.99% in this implementation, whereas C5 achieved an accuracy of 84.71%. Based on these results, C5 was selected as the machine learning algorithm to be implemented in the HEVC encoder.

IV. Online Learning-Based HEVC Encoder

The method proposed in this paper reduces HEVC encoding computational cost by limiting the maximum quadtree depth according to CU splitting decisions taken by a set of decision tree models. The C5 algorithm was implemented in the HM software to train the three decision tree models, one for each CU size that allows splitting, during encoding the time. As the analysis presented in the previous section has shown, training the model specifically for a video sequence based on its first frames allows better adaptation of the trained model to the content being encoded in the scene, which leads to higher decision accuracy than the obtained with a generic model trained offline for a diverse set of sequences.

Fig. 2 presents the video encoding flow employed by the proposed method. The first *M* frames of each video are encoded normally with no computational cost reduction, using the full RDO process to optimally define all partitioning structures. The value of *M* corresponds to half a second of playback and thus depends on the video frame rate. The half-a-second interval was decided through an empirical experiment varying the number of frames considered for this phase. For each 64×64, 32×32 and 16×16 CU encoded in the first *M* frames, the 10 attributes selected in section III.A are collected and stored for use in the model training process. After that, a pre-processing is performed over the data to resample the obtained instances aiming at reducing any possible class data imbalance problem. This problem occurs when there are significantly more training instances belonging to one class, biasing the models towards one specific decision. To solve that, training instances are resampled to yield a dataset with 50% of *Split* CUs and 50% of *Don't Split* CUs.

After the pre-processing step, the C5 algorithm is executed three times, once for each dataset (64×64, 32×32 and 16×16 CUs), and the obtained decision trees are stored for use in the remaining *N* frames. For each possible CU under evaluation in the *N* remaining frames, the 10 attributes are obtained and fed to the corresponding decision tree, which yields as outcome the *Split* or *Don't Split* classification. If the outcome

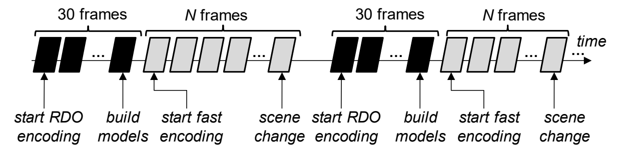

Fig. 2. Video encoding flow employed by the proposed method.

is *Split*, the recursive quadtree splitting process continues for the current CU; otherwise, it halts. The C5 algorithm was configured with different values of *m*, depending on the model, which represents the minimum number of instances classified in each decision tree node. The *m* parameter was set to 30, 150 and 350 for the 64×64, 32×32 and 16×16 models, respectively. These values were decided through an extensive empirical experiment varying the *m* parameter between 30 and 750 in steps of 25.

It is important to highlight that the method proposed in this paper is strongly based on the temporal correlation between frames within a video scene. Most typical scenes are composed of several seconds, so that the trained models are quite accurate during this period. However, whenever the scene changes the trained models become outdated. This way, the proposed method needs to be cyclically executed, training new models periodically or whenever a scene change is detected. A simple solution is to train a new model after a fixed number of frames is encoded based on the decision models. Algorithms for scene change detection are abundant in the literature and may also be integrated to the proposed solution and used to trigger new data collection and training phases whenever required. However, the discussion of such algorithms is not within the scope of this paper.

V. Experimental Results

The method proposed in this paper was evaluated in terms of computational cost reduction and compression efficiency for a set of video sequences different from those used in the data mining and algorithm selection phases. As explained in section IV, the C5 algorithm, the structure for decision trees and the mechanism to enable and disable splitting CUs were implemented in the HM encoder. Computational cost and compression efficiency were compared to values observed when encoding with the original HM implementation.

Experiments were performed using 10 video sequences of different resolutions, composed of 300 frames: *BQSquare*, *BQTerrace*, *BasketballDrill*, *BasketballPass*, *Cactus*, *ChinaSpeed*, *Kimono1*, *PeopleOnStreet*, *SlideEditing*, and *SteamLocomotive*. All video sequences were encoded with QPs 22, 27, 32, and 37, using the *Random Access Main* profile defined in CTC [15]. A workstation equipped with an Intel Core i7 (2.0 GHz) processor, 8GB of RAM and the Ubuntu 15.4 operating system was used in all tests. The value of *M* and *N* were set to 30 and 270 frames, respectively.

Table III presents average results in terms of Bjontegaard Delta bitrate (BD-rate) [17] and computational cost reduction (CCR) for the method proposed in this paper. BD-rate corresponds to the bitrate difference between two encoding solutions given the same image quality, so that a positive BD-rate value indicates bitrate increase (i.e., loss of compression efficiency) [17]. The table also presents the BD/CCR ratio (multiplied by 100), which indicates the compression efficiency loss of the proposed solution for each percentage point in CCR. The results in Table III show that the method is able to reduce computational cost in 34.4% on average, with a negligible compression efficiency loss of only 0.2%.

Table IV presents a comparison between this work and the best related strategies found in the literature that focus on computational cost reduction for the quadtree splitting process [5, 6, 7, 11, 12]. Even though average CCR results show that the method proposed in this work achieves the lowest CCR, BD-rate values show that this is by far the work with smallest impact in compression efficiency. Since each work achieved different levels of CCR, a fair comparison between all related works is only possible when the BD/CCR ratio is considered. Notice that among all comparable works, the proposed method presents the best tradeoff between compression efficiency and CCR.

TABLE III. EXPERIMENTAL RESULTS FOR THE PROPOSED METHOD.

Video Sequence	BD-rate (%)	CCR (%)	BD/CCR (× 100)
BQSquare	0.158	37.8	0.42
BQTerrace	0.002	36.6	0.00
BasketballDrill	0.167	27.6	0.60
BasketballPass	0.328	26.0	1.26
Cactus	0.053	21.2	0.25
ChinaSpeed	0.030	19.0	0.16
Kimono1	0.276	42.2	0.65
PeopleOnStreet	0.129	26.5	0.49
SlideEditing	0.341	54.2	0.63
SteamLocomotive	0.519	52.6	0.99
Average	**0.200**	**34.4**	**0.55**

TABLE IV. COMPARISON WITH RELATED WORKS.

	Seungh. [5]	Jong-H. [6]	Goswami [7]	Shen [11]	Correa [12]	Proposed
BD-rate (%)	0.6	1.2	1.7	1.4	0.28	**0.2**
CCR (%)	50	48	38	43	36.7	**34.4**
BD/CCR (× 100)	1.2	2.5	4.42	3.11	0.77	**0.55**

VI. CONCLUSIONS

This paper presented a fast HEVC encoding scheme employs decision tree models trained during the first steps of the encoding process to accelerate frame partitioning structure decisions in the remaining frames. As the models are specifically trained for each video scene, which are usually composed of several frames with high temporal correlation, they achieve very high decision accuracy. The paper also presented a thorough analysis over a set of candidate attributes and machine learning algorithms, aiming at finding those that yield the best decision accuracy levels and consequently the smallest impacts in compression efficiency when reducing the encoding computational cost. The C5 algorithm was implemented in the HM encoder and used to train three decision trees models during encoding time, which led to a computational cost reduction in the HEVC encoder that ranges from 19% up to 54%, at the cost of an average bit rate increase of only 0.2%. When compared to the best related works found in the literature, the proposed method achieved the best tradeoff between computational cost reduction and compression efficiency.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), and the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul, Brasil (FAPERGS).

REFERENCES

[1] International Telecommunication Union. Recommendation ITU-T H.265: High efficiency video coding. April, 2015.

[2] High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Encoder Description, JCTVC-T1002, ISO/IEC-JCT1/SC29/WG11, Geneva, 2015.

[3] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, “Performance and Computational Complexity Assessment of High Efficiency Video Encoders,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, pp. 1899-1909, 2012.

[4] J. Vanne, M. Viitanen, T. Hamalainen, and A. Hallapuro, “Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs,” Circuits and Systems for Video Technology, IEEE Transactions on, vol. 22, pp. 1885-1898, 2012.

[5] C. Seunghyun and K. Munchurl, “Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding,” IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 9, pp. 1555-1564, 2013.

[6] L. Jong-Hyeok, P. Chan-Seob, and K. Byung-Gyu, “Fast coding algorithm based on adaptive coding depth range selection for HEVC,” in IEEE Int. Conf. Cons. Electronics - Berlin, 2012, pp. 31-33.

[7] K. Goswami et al., “Early Coding Unit (CU) Splitting Termination Algorithm for High Efficiency Video Coding (HEVC),” Electronics and Telecommunications Research Institute Journal, to be published.

[8] G. F. Escribano, P. Cuenca, L. O. Barbosa, and H. Kalva, “Very low complexity MPEG-2 to H.264 transcoding using machine learning,” in ACM Int. Conf. Multimedia, Santa Barbara, 2006, pp. 931-940.

[9] L. P. Van et al., “Fast Transrating for High Efficiency Video Coding based on Machine Learning,” in IEEE Int. Conference on Image Proc., 2013, pp. 1573-1577.

[10] C. H. Lampert, “Machine learning for video compression: Macroblock mode decision,” in 18th International Conf. on Pattern Recognition, 2006, pp. 936-940.

[11] X. Shen and L. Yu, “CU splitting early termination based on weighted SVM,” EURASIP Journal on Image and Video Processing, vol. 2013, no. 4, pp. 1-11, 2013.

[12] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, “Fast HEVC encoding decisions using data mining,” IEEE Trans. on Circuits and Systems for Video Technology, v. 25, n. 4, p. 660-673, 2015.

[13] Rulequest Research. “Is See5/C5.0 Better Than C4.5?”. Available in <http://rulequest.com/see5-comparison.html>

[14] M. Hall, et al., “The WEKA data mining software: an update,” ACM SIGKDD Explorations Newsletter, vol. 11, pp. 10-18, 2009.

[15] Common test conditions and software reference configurations, JCTVC-J1100, ISO/IEC-JCT1/SC29/WG11, Stockholm, Sweden, 2012.

[16] J. R. Quinlan, C4.5: Programs for Machine Learning, San Francisco: Morgan Kaufmann Publishers, 1993.

[17] G. Bjontegaard, “Calculation of average PSNR differences between RD-curves,” Austin, TX, 2001.