Classification-Based Early Termination for Coding Tree Structure Decision in HEVC

[1,2] Guilherme Correa, [1,3] Pedro Assuncao,
[1,2] Luis A. da Silva Cruz
[1] Instituto de Telecomunicações
[2] University of Coimbra, Coimbra, Portugal
[3] Polytechnic Institute of Leiria, Leiria, Portugal
guilherme.correa@co.it.pt, amado@co.it.pt, luis.cruz@co.it.pt

Luciano Agostini
Group of Architectures and Integrated Circuits
Federal University of Pelotas
Pelotas, Brasil
agostini@inf.ufpel.edu.br

***Abstract*— The High Efficiency Video Coding (HEVC) standard provides improved compression rates in comparison to its predecessors at the cost of large increases in computational complexity. An important share of such increases is due to the introduction of flexible Coding Tree structures, which best configuration is decided through exhaustive tests in a Rate-Distortion Optimization (RDO) scheme. In this work, an early termination method for the decision of such structures was designed using classification trees obtained through Data Mining techniques. The classification trees were trained using intermediate encoding results from a training set of video sequences and implemented in the encoder to skip the full RDO-based decision. An average reduction of 37% in the HEVC encoder computational complexity was achieved when using the designed classification trees, with a negligible cost of only 0.28% in terms of Bjontegaard Delta-rate increase.**

Keywords—early termination; classification trees; data mining; computational complexity; high efficiency video coding (HEVC).

I. INTRODUCTION

The *High Efficiency Video Coding* (HEVC) standard, which has been recently launched by the *Joint Collaborative Team on Video Coding* (JCT-VC), achieved significant compression gains in comparison to its predecessor, the H.264/AVC standard. However, the improved compression efficiency of HEVC is obtained at the expense of significant increases in computational complexity, mainly resulting from much more intensive processing tools, nested partitioning structures and optimization algorithms dealing with larger amounts of data. According to [1], the computational complexity of HEVC is said to be 40% higher than that of H.264/AVC when only the essential tools are enabled.

The introduction of more flexible partitioning structures, namely the *Coding Tree Units* (CTUs), *Coding Units* (CUs), *Prediction Units* (PUs) and *Transform Units* (TUs), 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 [2]. In the *HEVC Test Model* (HM) [3], each CTU can be divided into several CUs following a recursive quadtree structure, the *Coding Tree*, which may assume variable depth, according to the encoder configuration. This flexible encoding structure allows the use of large CUs when large, homogenous regions of a frame are encoded and the use of small CUs in regions with a more detailed texture. This process, though yielding optimal compression efficiency, greatly increases the encoder's computational complexity, limiting its use in computationally and energy-constrained environments.

Some authors have recently proposed heuristics for simplifying the *Coding Tree* decision process, aiming at decreasing the computational complexity of HEVC [4-8]. However, even though these works succeed in their goal of decreasing the encoding complexity up to a certain extent, the majority of them come with associated non-negligible losses in terms of Rate-Distortion (R-D) efficiency. To overcome this problem, a few approaches which apply machine learning techniques for computational complexity reduction have been proposed in [9-11]. However, the method in [9] cannot be used in HEVC encoders because it was designed for previous standards that do not use *Coding Tree* structures, and the method in [10] does not take advantage of image characteristics and/or intermediate encoding results when training the model, which limits the encoder's R-D efficiency. The CU splitting early termination proposed in [11] makes use of features extracted during offline encodings and is able to reduce the computational complexity in reasonable amounts, but the work proposed in this paper yields better compression efficiency than [11], as shown later.

The approach proposed in this paper uses *Data Mining* (DM) as a tool to build a set of classification trees that allow early terminating the decision process that finds the best *Coding Tree* for a CTU, relieving the encoder of the complex task of testing all possibilities available. The classification trees designed using DM were trained in offline encodings and implemented in the HM encoder, leading to computational complexity reductions that range from 23% to 71% in comparison to the original encoder at the cost of negligible bit rate increases. The results outperform previous works both in terms of complexity savings and compression efficiency.

The rest of the paper is organized as follows. Motivations and a Rate-Distortion-Complexity (R-D-C) analysis that led to the development of the proposed method are presented in section II. The methodology and approach proposed in this work are detailed in section III. Experimental results and comparisons with related works are presented in section IV. Finally, section V concludes this paper.

This work was supported by IT-Portugal, CNPq-Brazil, and the project FCT-CAPES (4.4.1.00 CAPES) (FCT/1909/27/2/2014/S) by FCT-Portugal and CAPES-Brazil.

II. Rate-Distortion-Complexity Analysis

This section presents an R-D-C analysis of the HEVC encoder when varying the maximum *Coding Tree* depth allowed when encoding a video. The observations performed during this analysis serve as motivation for the development of the method presented in this paper.

In the HM encoder, *Max CU Depth* is the encoding parameter that controls directly the maximum *Coding Tree* depth allowed for each CTU in a frame. The R-D-C analysis was performed by encoding 10 video sequences with different spatial and temporal resolution (*BlowingBubbles*, *RaceHorses*, *PartyScene*, *BQMall*, *SlideShow*, *vidyo1*, *BasketballDrive*, *ParkScene*, *NebutaFestival*, *Traffic*) using four different encoding configurations, which differ from one another in the value of *Max CU Depth* (from 1 to 4). These configurations are called here as CFG_n, where *n* represents the maximum *Coding Tree* depth specified by *Max CU Depth*. The *Random Access* (RA) temporal configuration was used in the analysis and the resulting image quality, bit rate and complexity were measured in terms of Bjontegaard-Delta (BD)-rate, BD-PSNR [12] and encoding time.

Table I shows the BD-rate increase and the average computational complexity reduction (CCR) per configuration, considering QPs 22, 27, 32 and 37. Both BD-rate and CCR values were calculated using CFG_4 as reference, which is the baseline case. An analysis of the results in Table I shows that computational complexity can be enormously reduced by changing the value of *Max CU Depth*. However, this reduction incurs in very high R-D costs in most cases (from 5.1% to 32.8%). To achieve a better trade-off between computational complexity and compression efficiency, it would be desirable to allow the encoder to choose the best *Max CU Depth* for each image region and video segment according to local characteristics. Indeed, instead of simply limiting the *Coding Tree* depth indiscriminately in the whole video sequence, the encoder should be able to decide which depth to use in a smaller scale, such as at a per-frame or per-CU scale, adapting itself to the video changing characteristics. This is the goal of the method presented in the following sections of this paper.

TABLE I
RATE-DISTORTION-COMPLEXITY ANALYSIS WHEN VARYING MAX CU DEPTH

Video Sequence	BD-rate (%)			CCR (%)		
	CFG_3	CFG_2	CFG_1	CFG_3	CFG_2	CFG_1
BlowingBubbles	5.2	21.9	41.3	25.2	49.1	70.3
RaceHorses	8.3	28.0	49.4	26.4	51.6	71.9
PartyScene	6.9	19.1	35.9	23.4	54.7	74.3
BQMall	7.2	23.0	45.3	23.2	55.1	73.4
SlideShow	12.8	29.5	52.1	22.4	52.6	73.1
vidyo1	2.6	11.4	27.3	21.7	52.4	73.4
BasketballDrive	1.4	8.3	21.0	21.6	54.1	75.9
ParkScene	3.5	11.4	24.4	20.1	52.5	75.0
NebutaFestival	0.2	1.2	4.1	27.4	59.7	80.2
Traffic	3.1	11.9	27.3	22.1	53.7	75.6
Average	**5.1**	**16.6**	**32.8**	**23.3**	**53.5**	**74.3**

III. Classification Trees for Coding Tree Decision

DM techniques are used to determine the value of dependent variables by looking at the value of some attributes in the data set, building generalization rules that are expressed as models. Classification trees are models acquired through predictive DM that are commonly used due to their low-complexity implementation. The approach presented in this article consists in building classification trees for accelerating the HEVC encoding process. More specifically, a set of trees are trained and implemented into HM to control the early termination of the recursive search for the best *Coding Tree*.

A. Methodology

The *Waikato Environment for Knowledge Analysis* (WEKA) [13], version 3.6, was used to aid the DM process performed in this work and the HM software (version 12) [3] was used to encode the video sequences with QPs 22, 27, 32, 37, and 42, using the RA temporal configuration. The 10 video sequences mentioned in section II were used to collect the data used for training the classification trees. In order to guarantee that the trained trees are generic enough to be used with any video sequence, the 10 video sequences selected for the training process differ broadly in terms of spatial and temporal resolution, as well as frame rate, texture motion activity.

When building the classification trees for the early termination proposed in this work, the usefulness of each attribute was evaluated through the *Information Gain Attribute Evaluation* (IGAE) method in WEKA, which measures the information gain [14] with respect to the class for each attribute. Based on this measure, the most relevant attributes were selected for the training process. IGAE was chosen for this evaluation because it is also used by the *C4.5* algorithm [14], which is the training algorithm used to build the classification trees.

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. When the best attribute is chosen, it is used to divide the training data into two sub-sets and then the same process is applied recursively. After trained, the accuracy of the obtained trees was observed with WEKA by applying a 10-fold cross-validation process. The level of accuracy was measured as the percentage of correct decisions in the total amount of instances used in the training process. Finally, the trees were implemented in HM.

B. Selecting Attributes and Training Classification Trees

The proposed early termination consists in deciding whether a CU should be split so that the encoder tests its four smaller sub-CUs. This decision is performed using the designed classification trees, which outcome is *yes* (i.e., *Split* = 1) or *no* (i.e., *Split* = 0). To reduce the problem of class data imbalance, which occurs when there are more training instances belonging to one class than to the other(s), following common practice [15], the training files are composed of a data set with 50% of CUs which have been split into smaller CUs and 50% of CUs which have not been split into smaller CUs.

As the HEVC standard allows up to four *Coding Tree* depths, three classification trees were trained, one for each size that allows splitting: 64×64, 32×32, and 16×16. Table II shows the variables that provided best performance as measured by the information gain with respect to the decision of splitting or not splitting a CU and were therefore selected as attributes for the classification trees. Information gain values are presented separately for each CU size in Table II and the most relevant attributes are detailed in the next paragraphs.

Partition represents the PU splitting mode that was chosen for the current CU among those available in HEVC (i.e., *2N×2N*, *2N×N*, *N×2N*, *N×N*, *2N×nU*, *2N×nD*, *nL×2N*, or *nR×2N*). This partition mode information is important, as it is likely that a CU predicted with large-sized PUs (e.g., as a single *2N×2N* PU) does not need being split into smaller CUs. Statistics that support this claim are presented in Fig. 1(a) for 32×32 CUs. The chart shows that 78.5% of the CUs encoded as a *2N×2N* PU were not split into sub-CUs. Conversely, an average of 86.7% of CUs encoded with the remaining PU splitting modes were divided into sub-CUs. The remaining CU sizes also show the behavior illustrated in Fig. 1(a).

Neigh_Depth is related with the *Coding Tree* depths used in neighboring CTUs already encoded. The rationale for considering this variable in the decision process is that there exists a correlation among maximum depths of spatially neighboring CTUs [5]. The value of *Neigh_Depth* is calculated as follows. First, for each neighboring CTU, the average depth of all CUs is computed. Then, the average of averages is computed among all neighbors. The top, left, top-left and top-right CTUs in the current frame, as well as the co-located CTUs in the first frames of both reference lists are considered as neighbors. Fig. 1(b) shows the distribution of *Neigh_Depth* for 32×32 CUs. It is clear from the data shown that most CUs with small *Neigh_Depth* are not split into smaller CUs. On the other hand, most CUs with *Neigh_Depth* above 1.63 are split.

The *Ratio(2N×2N, MSM)* is calculated as a simple division between the R-D costs of encoding the current CU as an inter-predicted *2N×2N* PU and as a *Merge/SKIP* (*MSM*) PU. The *relRatio(2N×2N, MSM)* value is the normalized difference of the *RD(2N×2N)* and *RD(MSM)* costs, calculated as per (1). The reason for considering these values in the IGAE analysis is that when a compression gain (i.e., a drop in R-D cost) is observed due to the use of motion compensated prediction in a CU instead of encoding it with *MSM*, the block probably belongs to a medium/high-motion or complex-textured image region and usually in this type of situation it is advisable to split a CU into smaller CUs.

$$relRatio(2N\times 2N, MSM) = \left| \frac{RD(2N\times 2N) - RD(MSM)}{RD(MSM)} \right| \quad (1)$$

The R-D costs for inter *2N×2N*, *MSM*, *2N×N* and *N×2N* PU splitting modes were also separately considered in the IGAE analysis and are represented in Table II. Also present in the table are the *MergeFlag* and *SkipMergeFlag*, which are binary variables used by the encoder to identify CUs that have been predicted with *MSM* and *SKIP* mode, respectively. These flags were included in the analysis because CUs encoded with *MSM* and *SKIP* modes generally belong to low-motion or very homogeneous image regions, which are rarely encoded with small CUs.

Table III presents performance indicators and topological characteristics of the obtained classification trees. The table presents the accuracy of each tree, as well as their depth (i.e., number of sequential tests), the number of test nodes and the number of leaves. As shown, the designed trees achieve very good decision accuracy, with values slightly larger than 84%. However, these results account the case of splitting a CU that should not be split into smaller CUs as an inaccuracy, even though it does not harm the encoding R-D efficiency. In fact, the R-D efficiency could only be harmed when a CU that should be split is not split due to the early termination. The third column of Table III shows the number of incorrect early terminations caused by inaccuracies in the classification trees, which are the cases that could actually cause R-D efficiency losses. It is also important to notice that all trees are composed of less than 10 levels (depth), which means that their associated computational complexity is negligible.

TABLE II
INFORMATION GAIN OF THE ATTRIBUTES IN EACH TREE

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

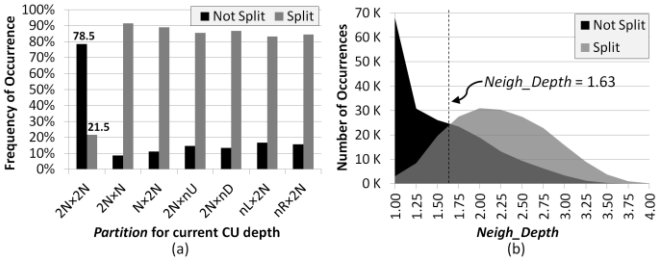

Fig. 1: Occurrence of CUs split and not split into smaller CUs according to (a) PU splitting mode chosen for the current CU and (b) average of CU depths in neighboring CTUs.

TABLE III
CHARACTERISTICS OF OBTAINED CLASSIFICATION TREES

CU size	Classifier Accuracy	Incorrect Depth	Depth	Test Nodes	Leaves
64×64	84.2%	7.1%	5	6	19
32×32	84.5%	7.5%	8	20	33
16×16	84.6%	6.9%	9	23	44

IV. EXPERIMENTAL RESULTS

In order to evaluate the performance of the early termination method, the three classification trees were implemented in the HM encoder (version 12) and a set of 10 video sequences with different spatial and temporal resolution (*BasketballPass*, *BQSquare*, *BasketballDrill*, *ChinaSpeed*, *Kimono1*, *SlideEditing*, *BQTerrace*, *Cactus*, *PeopleOnStreet*, *SteamLocomotive*) was used to validate the proposed method. Notice that none of the sequences used in the evaluation was included in the training of the trees. Bit rate, PSNR and encoding times were compared with the values obtained with the unmodified HM encoder and all video sequences were encoded with QPs 22, 27, 32, and 37, using the *Main* profile and the baseline RA temporal configuration (i.e., *Max CU Depth* set to 4). PSNR was calculated as the average luminance PSNR (Y-PSNR) among all frames in each video sequence. It is important to notice that the early termination method proposed in this paper was implemented on top of all computational complexity reduction techniques already

available in HM, providing additional reductions. All built-in complexity reduction methods of HM were also enabled in the original version of HM used for comparisons.

Table IV presents R-D-C results for the proposed early termination method. The table shows that an average CCR of 37% is achieved with negligible R-D efficiency losses. Notice that, even though the CCR achieved with the proposed method is comparable to those achieved with configurations CFG_2 and CFG_3 in section II (53.5% and 23.3%, respectively), the associated BD-rate increase is much smaller in this case. While those configurations incurred in BD-rate increases of 5.1% and 16.6%, the use of the proposed classification trees incurs in a BD-rate increase of only 0.28%, which is 18 and 59 times smaller than the incurred by CFG_2 and CFG_3, respectively.

The rightmost column of Table IV shows the ratio between BD-rate and CCR, which is useful for comparisons with similar works that yield different CCR levels. A comparison with such works is presented in Table V. Notice, however, that the compared works were not all tested under the exact same conditions, since each author implemented its method in a different HM version and tested it with a different set of video sequences. Still, it is possible to notice that on average the BD-rate increase relative to CCR is smaller in the method proposed in this paper. While the related works present BD-rate/CCR ratios varying from 1.20 to 4.42, the proposed early termination presents a ratio of only 0.76, which means that it offers a much more efficient Rate-Distortion-Complexity trade-off.

TABLE IV
RATE-DISTORTION-COMPLEXITY RESULTS FOR THE EARLY TERMINATION

Video Sequence	BD-rate (%)	BD-PSNR (dB)	CCR (%)	BD-rate / CCR
BQSquare	-0.01	0.00	23	-0.03
BQTerrace	-0.01	0.00	28	-0.03
BasketballDrill	+0.43	-0.02	30	1.41
BasketballPass	+0.13	-0.01	24	0.53
Cactus	+0.21	-0.01	41	0.51
ChinaSpeed	+0.13	-0.01	29	0.45
Kimono1	+0.55	-0.02	44	1.25
PeopleOnStreet	+0.09	0.00	16	0.56
SlideEditing	+0.35	-0.06	71	0.50
SteamLocomotiveTrain	+0.97	-0.02	60	1.61
Average	**+0.28**	**-0.01**	**37**	**0.76**

TABLE V
COMPARISON WITH RELATED WORKS

Related Work	BD-rate (%)	CCR (%)	BD-rate / CCR
Seunghyun [4]	+0.6	50	1.20
Correa [5]	+0.8	40	2.00
Jong-Hyeok [6]	+1.2	48	2.50
Goswami [7]	+1.7	38	4.47
Jian [8]	+1.9	43	4.42
Shen [11]	+1.4	45	3.11
Proposed	**+0.28**	**37**	**0.76**

V. CONCLUSIONS

This paper presented an early termination method that reduces the computational complexity of the HEVC encoder. The method was developed making use of *Data Mining* tools to build classification trees that exploit intermediate encoding results and halt the process that decides the best *Coding Tree* structure for each CTU. The classification trees were trained and validated through extensive experiments using distinct sets of video sequences in each phase. Experimental results have shown that an average complexity reduction of 37% can be achieved when the early termination method is used, with a BD-rate increase of only 0.28%, yielding a better R-D-C efficiency than the best related works published so far. The proposed method does not add any computationally intensive operations to the HEVC encoder, since the classification trees use only intermediate encoding results.

REFERENCES

[1] J. Vanne, M. Viitanen, T. Hamalainen, and A. Hallapuro, "Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. PP, pp. 1-1, 2012.

[2] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC Complexity and Implementation Analysis," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, pp. 1685-1696, 2012.

[3] ISO/IEC-JCT1/SC29/WG11, "High Efficiency Video Coding (HEVC) Test Model 12 (HM 12) Encoder Description," ed. Vienna, Austria, 2013.

[4] C. Seunghyun and K. Munchurl, "Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 23, pp. 1555-1564, 2013.

[5] G. Correa, P. Assuncao, L. Agostini, and L. Silva Cruz, "Complexity scalability for real-time HEVC encoders," *Journal of Real-Time Image Processing,* pp. 1-16, 2014/01/05 2014.

[6] L. Jong-Hyeok, P. Chan-Seob, and K. Byung-Gyu, "Fast coding algorithm based on adaptive coding depth range selection for HEVC," in *Consumer Electronics - Berlin (ICCE-Berlin), 2012 IEEE International Conference on*, 2012, pp. 31-33.

[7] Kalyan Goswami, Byung-Gyu Kim, Dong-San Jun, Soon-Heung Jung, and J. S. Choi, "Early Coding Unit (CU) Splitting Termination Algorithm for High Efficiency Video Coding (HEVC)," *to be published in ETRI Journal,* 2014.

[8] X. Jian, L. Hongliang, W. Qingbo, and M. Fanman, "A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence," *Multimedia, IEEE Transactions on,* vol. 16, pp. 559-564, 2014.

[9] C. H. Lampert, "Machine learning for video compression: Macroblock mode decision," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006, pp. 936-940.

[10] Ray Garcia, Damian Ruiz Coll, Hari Kalva, and G. Fernandez-Escribano, "HEVC Decision Optimization for Low Bandwidth in Video Conferencing Applications in Mobile Environments " in *IEEE International Conference on Multimedia and Expo (ICME 2013)*, San Jose, USA, 2013.

[11] X. Shen and L. Yu, "CU splitting early termination based on weighted SVM," *EURASIP Journal on Image and Video Processing,* vol. 2013, p. 4, 2013.

[12] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," ed. Austin, Texas, 2001.

[13] M. Hall*, et al.*, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.,* vol. 11, pp. 10-18, 2009.

[14] J. R. Quinlan, *C4.5: Programs for Machine Learning*: Morgan Kaufmann Publishers, 1993.

[15] N. Japkowicz, "The Class Imbalance Problem: Significance and Strategies," in *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-{AI}'2000)*, 2000, pp. 111-117.