Analog Integr Circ Sig Process (2016) 87:129–139
DOI 10.1007/s10470-016-0719-z

Fast coding tree structure decision for HEVC based on classification trees

Guilherme Correa[1] · **Pedro Assuncao**[2,3] · **Luciano Agostini**[1] · **Luis A. da Silva Cruz**[2,4]

Received: 30 April 2015 / Revised: 8 March 2016 / Accepted: 12 March 2016 / Published online: 18 March 2016

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 set of video sequences and implemented in the encoder to avoid 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.

✉ Guilherme Correa
gcorrea@inf.ufpel.edu.br

Pedro Assuncao
amado@co.it.pt

Luciano Agostini
agostini@inf.ufpel.edu.br

Luis A. da Silva Cruz
luis.cruz@co.it.pt

[1] Group of Architectures and Integrated Circuits, Federal University of Pelotas, Pelotas, Brazil

[2] Instituto de Telecomunicações, Lisbon, Portugal

[3] Polytechnic Institute of Leiria, Leiria, Portugal

[4] University of Coimbra, Coimbra, Portugal

Keywords Early termination · Coding trees · Data mining · Classification trees · Computational complexity · High efficiency video coding (HEVC)

1 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 authors in [2] claim that this complexity increase can achieve 500 % if all encoding tools are enabled.

Until recently, most research works have focused on the inter-frame prediction process in order to reduce the encoding complexity of H.264/AVC and previous standards, since this is considered the most complex operation of those standards [3–6]. However, the decision of the new partitioning structures introduced in HEVC increased significantly the encoding computational complexity, so that part of the research effort has been shifted to this task. These flexible partitioning structures, namely the coding tree units (CTUs), the coding units (CUs), the prediction units (PUs) and the transform units (TUs), have been claimed as the main responsible for the compression gains achieved by HEVC, but have also been charged as the standard's most computationally demanding inclusions [7]. In the *HEVC Test Model* (HM) [8], 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 [9–13]. 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 [14–16]. However, the method in [14] cannot be used in HEVC encoders because it was designed for previous standards that do not use *Coding Tree* structures, and the method in [15] 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 [16] 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 article yields better compression efficiency than [16], as shown later.

The approach proposed in this article 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 article is organized as follows. Section 2 describes the partitioning structures introduced in HEVC. A rate-distortion-complexity (R-D-C) analysis that motivates the development of the method proposed in this article are presented in Sect. 3. The research methodology and proposed approach are detailed in Sect. 4. Experimental results, method robustness analysis and comparisons with related works are presented in Sect. 5. Finally, Sect. 6 concludes this article.

2 HEVC partitioning structures

In HEVC, each frame may be divided into a set of *slices*, which are parts of the frame that can be independently decoded. A slice is composed of sequential CTU, which are composed of one luminance *Coding Tree Block* (CTB) of size $W \times W$ and two chrominance CTBs of size $W/2 \times W/2$, where W may be equal to 8, 16, 32 or 64. The luminance CTB together with the chrominance CTBs and associated syntax elements form a CTU, which is the basic image data unit of HEVC. As the same partitioning structure decisions are applied to luminance CTB and chrominance CTBs, from now on this article will only refer to CTUs and their sub-divisions.

Each CTU can be divided into smaller blocks, called CU, following a recursive quadtree structure, the *Coding Tree*. The CTU is the root of the *Coding Tree*, which may assume variable depth, according to the encoder configuration. The *Largest Coding Unit* (LCU) size and the *Smallest Coding Unit* (SCU) size used by the encoder are defined in its configuration, but the maximum and minimum allowed sizes for a CU are 64×64 and 8×8, respectively, so that up to four *Coding Tree* depths are possible. In the HM encoder, the *Coding Tree* structure is defined by an iterative splitting process, which evaluates all possibilities in a Rate-Distortion Optimization (RDO) scheme, until the SCU depth is reached. Figure 1 shows an example of a 64×64 CTU divided into several CUs. The example shows the final *Coding Tree* division chosen after all possibilities are evaluated. The tree leaves (grey blocks) are the final CUs in the coding quadtree. 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.

For intra and inter-frame prediction, each CU may be divided into two or four PUs, which are separately

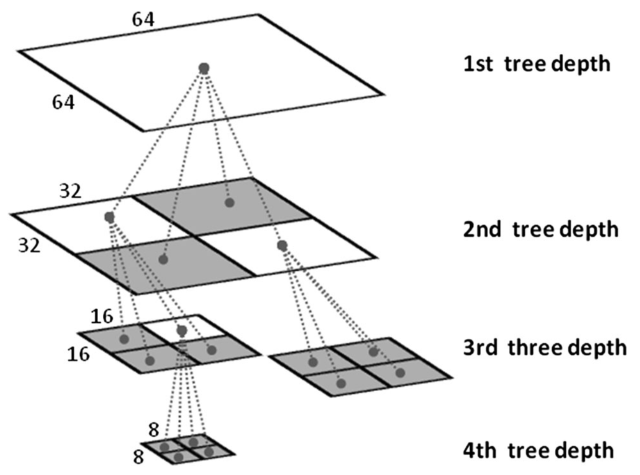


Fig. 1 Coding Tree structure of a CTU divided into CUs

predicted. All PUs in a CU are predicted with either inter-frame or intra-frame prediction. Figure 2 presents all possible PU splitting modes that can be used in the prediction of a CU. In HM, the best PU splitting mode is chosen using an RDO process, which evaluates and compares all possibilities in terms of bit rate and objective quality.

The number of supported PU splitting modes varies according to the CU size. Inter-predicted $N \times N$ PUs are only tested if the CU is a LCU, since in the remaining cases the same area is tested as a $2N \times 2N$ PU in the upper depth of the *Coding Tree*. Inter-predicted $N \times N$ PUs are not tested in 8×8 CUs to reduce memory bandwidth requirements. *Asymmetric Motion Partitions* (AMP), i.e., $2N \times nU$, $2N \times nD$, $nL \times 2N$, and $nR \times 2N$ [17], are not tested in 8×8 CUs to prevent the use of PUs with dimensions smaller than 4. Finally, only the *Square* modes (i.e., $2N \times 2N$ and $N \times N$) are used in intra-predicted PUs.

The *Merge/SKIP Mode* (*MSM*) is available for all inter-predicted CU sizes and is conceptually similar to the *SKIP* mode of H.264/AVC. Although considered a different PU splitting mode, *MSM* is applied to $2N \times 2N$ PUs, so that it can be seen as a sub-mode of inter $2N \times 2N$. When *MSM* is used, the motion information from spatially and temporally neighboring PUs is inherited by the current PU, forming a larger merged region. The *SKIP* mode is treated as a special case of *MSM*.

3 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. This *Coding Tree* structure was chosen for the analysis because it was found to be

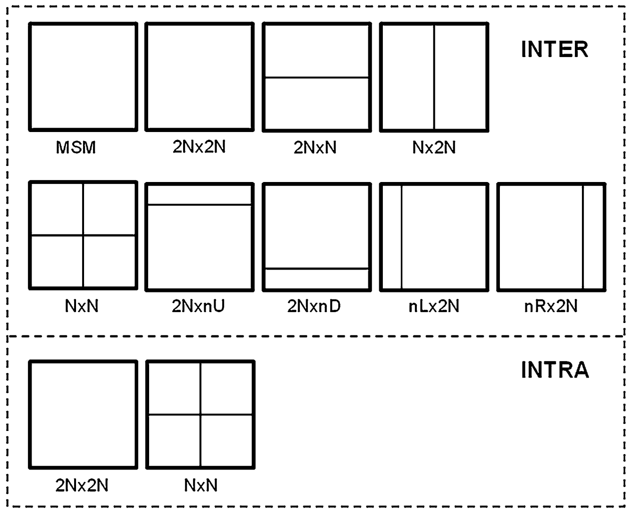


Fig. 2 Inter-frame and intra-frame PU splitting modes available in HEVC

one of the main reasons for the high computational complexity of HEVC [18]. The observations performed during this analysis serve as motivation for the development of the method presented throughout this article.

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 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 [19] was used in the analysis and the resulting image quality, bit rate and complexity were measured in terms of Bjontegaard-Delta (BD)-rate [20] and encoding time. The 10 video sequences used in these experiments are those in Table 1 that belong to the *A* set (sequences that belong to the *B* set are used later in this article).

Table 2 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 2 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 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-CTU scale, adapting itself to the video changing characteristics. This is the goal of the method presented in the following sections of this article.

4 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*.

Table 1 Test sequences used for analysis, training and validation

Video sequence	Frame rate (Hz)	Bit depth	Spatial resolution	Set
Blowing Bubbles	50	8	416 × 240	A
RaceHorses	30	8	416 × 240	A
PartyScene	50	8	832 × 480	A
BQMall	60	8	832 × 480	A
SlideShow	20	8	1280 × 720	A
vidyo1	60	8	1280 × 720	A
BasketballDrive	50	8	1920 × 1080	A
ParkScene	24	8	1920 × 1080	A
NebutaFestival	60	10	2560 × 1600	A
Traffic	30	8	2560 × 1600	A
BasketballPass	50	8	416 × 240	B
BQSquare	60	8	416 × 240	B
BasketballDrill	50	8	832 × 480	B
ChinaSpeed	30	8	1024 × 768	B
Kimono1	24	8	1024 × 768	B
SlideEditing	30	8	1280 × 720	B
BQTerrace	60	8	1920 × 1080	B
Cactus	50	8	1920 × 1080	B
PeopleOnStreet	30	8	2560 × 1600	B
SteamLocomotive	60	10	2560 × 1600	B

Table 2 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

4.1 Methodology

The *Waikato Environment for Knowledge Analysis* (WEKA) [21], version 3.6, was used to aid the DM process performed in this work and the HM software (version 12) [8] was used to encode the video sequences with QPs 22, 27, 32, 37, and 42, using the RA temporal configuration [19]. The 10 video sequences mentioned in Sect. 3 (i.e., those that belong to the *A* set in Table 1) 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, these 10 video sequences 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 [22] achievable by using a variable to classify the data into the different classes represented in the data. 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. Therefore, the information gain of a variable indicates how relevant it is for the process of constructing a decision tree that correctly decides to which class each data item belongs. In the case of the WEKA software, this information gain is measured by the Kullback–Leibler divergence (KLD) [23] of the pre and post-classification probability distributions. Based on this measure, a manual analysis procedure was followed to identify the most useful variables for the (tree) decision processes. Then, the variables with higher information gain were selected as attributes for the tree training processes, as explained later. IGAE was chosen for this evaluation because it is also used by the *C4.5* algorithm [22], 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 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. 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.

4.2 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 [24], 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 3 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. The values presented in Table 3 refer to the information gain achieved with each variable considering each CU size. The variables are described in the next paragraphs.

The *Partition* attribute corresponds to which PU splitting mode was chosen for the current CU (i.e., $2N \times 2N$, $2N \times N$, $N \times 2N$, $N \times N$, $2N \times nU$, $2N \times nD$, $nL \times 2N$, or $nR \times 2N$), independently of whether inter or intra-frame prediction was applied. The idea behind saving this information is that when a large PU (e.g., $2N \times 2N$) is chosen as the best option to predict a determined CU, further tests to determine the *Coding Tree* configuration are probably not necessary, so that this CU does not need being split into smaller sub-CUs. Statistics that support this claim are presented in Fig. 3. The chart shows that most of the CUs predicted as a $2N \times 2N$ PU were not split into sub-CUs. For

Table 3 Information gain of the attributes in each tree

Attribute	64 × 64	32 × 32	16 × 16
Partition	0.352	0.336	0.269
ΔNeighDepth	0.311	0.262	0.249
Ratio(2N × 2N, MSM)	0.112	0.168	0.255
NormDiffRD(2N × 2N, 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

example, 83 % of 64 × 64 CUs predicted as a $2N \times 2N$ PU did not need being split into four 32 × 32 CUs, as the leftmost black bar of Fig. 3(a) shows. Conversely, an average of 83.3 % of 64 × 64 CUs encoded with the remaining modes were split into four 32 × 32 CUs (average of all grey bars of Fig. 3(a), except for the $2N \times 2N$ case). By analyzing the three charts of Fig. 3, it is possible to notice that, on the one hand, the correlation between using $2N \times 2N$ PUs and not splitting the CU decreases in smaller CUs (70 % for 16 × 16 CUs, as shows Fig. 3(c)). On the other hand, in smaller CUs the correlation between choosing the remaining PU modes and splitting the CU increases (on average, 90 % for 16 × 16 CUs).

The *ΔNeighDepth* attribute is computed based on the difference between the *Coding Tree* depths used in neighboring CTUs and the depth of the current CU. The rationale of considering such variable is that there exists a strong correlation among maximum depths of spatially and

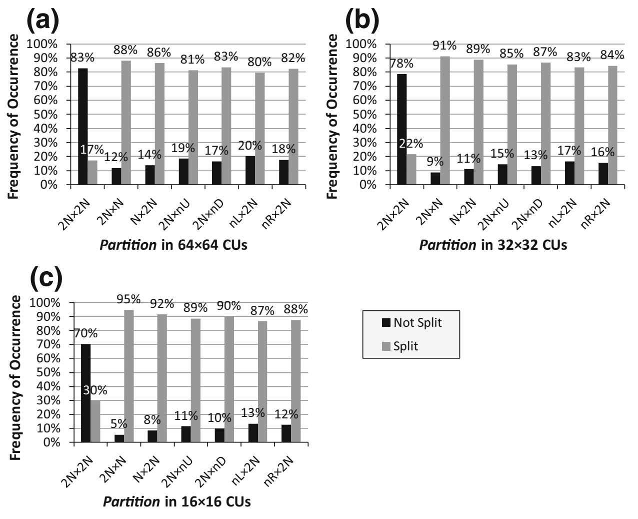


Fig. 3 Occurrence of **a** 64 × 64 CUs, **b** 32 × 32 CUs, and **c** 16 × 16 CUs split and not split into smaller CUs according to the PU mode chosen (i.e., Partition)

temporally neighboring CTUs, as shown in [10]. The attribute *ΔNeighDepth* is calculated as follows. First, for each neighboring CTU, the average depth among all its composing CUs is computed. The top-left, top, top-right and left CTUs in the current frame, as well as the co-localized CTUs in the first frames of both reference lists (*List 0* and *List 1*) are considered as neighbors, so that up to six average depths are calculated. Figure 4 shows the four neighboring CTUs in the current frame and the co-localized CTUs in the reference frames, with the variables representing the average CU depths assigned to them. Let us call these averages as A_1 to A_N, where N is the number of neighboring CTUs available for the current CTU. The CTU assigned with label C in Fig. 4 represents the current CTU. Finally, the value of *ΔNeighDepth* is calculated as an average of the averages A_1 to A_N, minus the depth of the CU currently being encoded, as shown in (1). If the current CU depth is much smaller than the average of average depths of neighboring CTUs, the splitting process should probably continue due to spatio-temporal correlation among neighboring CTUs.

$$\Delta NeighDepth = \frac{\sum_{i=1}^{N} A_i}{N} - CurrDepth \tag{1}$$

Figure 5 shows the distribution of *ΔNeighDepth* for different CU sizes. The curves show that there is a clear relationship between the distribution of *ΔNeighDepth* and the CU splitting decision. CUs that are not split into smaller CUs have *ΔNeighDepth* values that cluster towards low magnitudes, while the opposite occurs for those CUs that are split into smaller CUs. Since the two distributions do not fully overlap, it is possible to determine an optimal decision threshold that minimizes the classification error rate. WEKA computes these thresholds during the process of training the decision trees.

The *Ratio(2N × 2N, MSM)* shown in Table 3 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 *MSM* PU, as shown in (2). The *NormDiffRD(2N × 2N, MSM)* value is the normalized difference between the *RD(2N × 2N)* and *RD(MSM)* costs, calculated as per (3).

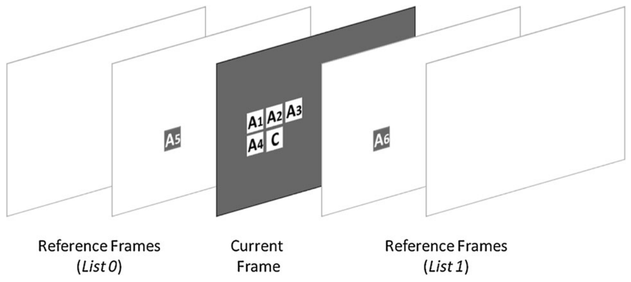


Fig. 4 Neighboring CTUs used in the calculation of ΔNeighDepth

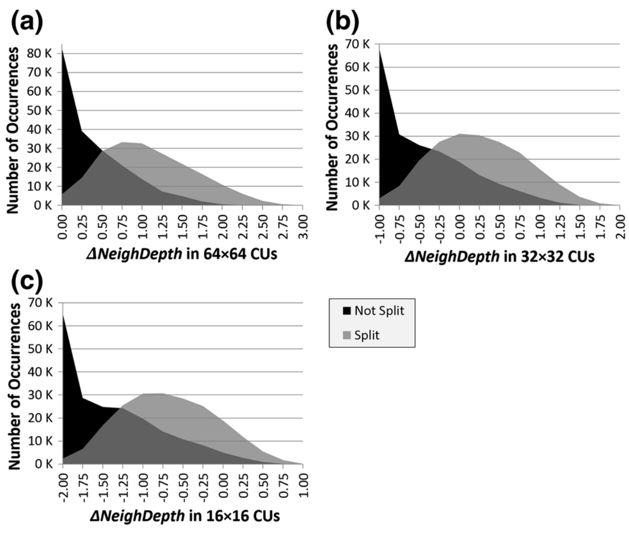


Fig. 5 Occurrence of **a** 64 × 64 CUs, **b** 32 × 32 CUs, and **c** 16 × 16 CUs split and not split into smaller CUs according to the average of CU depths in neighboring CTUs (i.e., ΔNeighDepth)

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. Figure 6 shows the distribution of *Ratio(2N × 2N, MSM)* for different CU sizes. The smaller information gain level of this parameter in comparison to the two previously analyzed cases is also clear in the charts, which shows the *Split* and *Not Split* areas more overlapped than in the charts of Figs. 3 and 5.

$$Ratio(2N \times 2N, MSM) = \frac{RD(2N \times 2N)}{RD(MSM)} \tag{2}$$

$$NormDiffRD(2N \times 2N, MSM) = \left| \frac{RD(2N \times 2N) - RD(MSM)}{RD(MSM)} \right| \tag{3}$$

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 3. 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 4 presents the topological characteristics of the obtained classification trees, including their depth, the number of test nodes and the number of leaves. It is

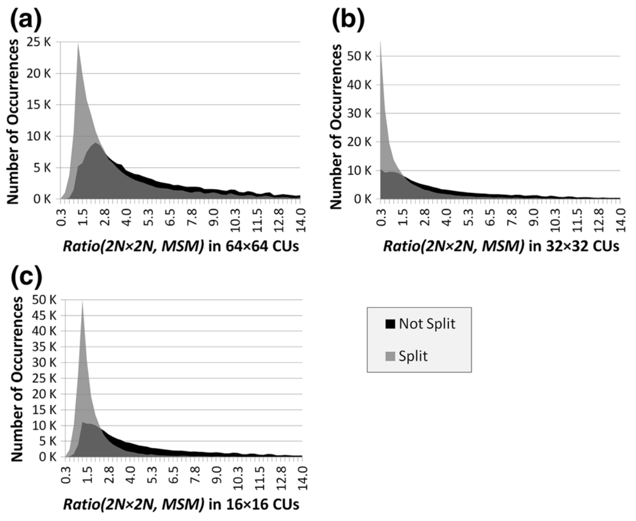


Fig. 6 Occurrence of **a** 64 × 64 CUs, **b** 32 × 32 CUs, and **c** 16 × 16 CUs split and not split into smaller CUs according to the ratio between the 2N × 2N and MSM R-D costs (i.e., Ratio(2N × 2N, MSM))

important to emphasize here that the *depth* in Table 4 refers to the number of decision levels in the obtained classification trees (i.e., the height of the classification tree) and does not have any relationship with the coding tree depth mentioned along this article. Table 4 shows that all classification trees are composed of <10 decision levels, which means that their associated computational complexity is negligible.

5 Experimental results

In order to evaluate the accuracy and the performance of the early termination method, the three classification trees presented in the previous section were implemented in the HM encoder (version 12) and a set of ten video sequences was used to validate the proposed method. In all tested cases, the *Random Access* (RA) temporal configuration [19] was used with QPs 22, 27, 32, and 37.

5.1 Classification accuracy and R-D-C efficiency analysis

To guarantee the trustfulness of this evaluation, the experiments described in this section use the ten video sequences that belong to the *B* set in Table 1, so that none of the sequences used in the analysis presented in Sect. 3 and in the training process presented in Sect. 4 was used here. The classification accuracy of each tree was measured as the percentage of CUs that were classified in the same class chosen by the original HM encoder (without any modification), i.e. *Split* or *Not Split*. Table 5 presents the classification accuracy of each tree. Notice that the designed trees achieve very good 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 5 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 loss due to choosing incorrect coding tree depths.

To illustrate the classification accuracy, Fig. 7 presents the 100th frame of the *BasketballDrill* video sequence and its corresponding CU boundaries according to the *Coding Tree* defined for each CTU. In this specific case, the sequence was encoded with QP 32. The frame Fig. 7(a) was encoded using the original HM encoder, while the frame in Fig. 7(b) was encoded with an HM encoder modified to include the *Coding Tree* early termination algorithm. It is possible to perceive that the differences between the boundaries in Fig. 7(a) and (b) are not expressive, which confirms that the early termination performs a correct *Coding Tree* determination in most cases.

Table 6 presents average 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 when the early termination method is implemented in the HM encoder. Notice that, even though the CCR achieved with the proposed method is between those achieved with configurations CFG_2 and CFG_3 presented earlier in Table 2 (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,

Table 4 Topological characteristics of obtained classification trees

CU size	Depth	Test nodes	Leaves
64 × 64	5	6	19
32 × 32	8	20	33
16 × 16	9	23	44

Table 5 Accuracy of obtained classification trees

CU size	Classifier accuracy (%)	Incorrect depth (%)
64 × 64	84.2	7.1
32 × 32	84.5	7.5
16 × 16	84.6	6.9

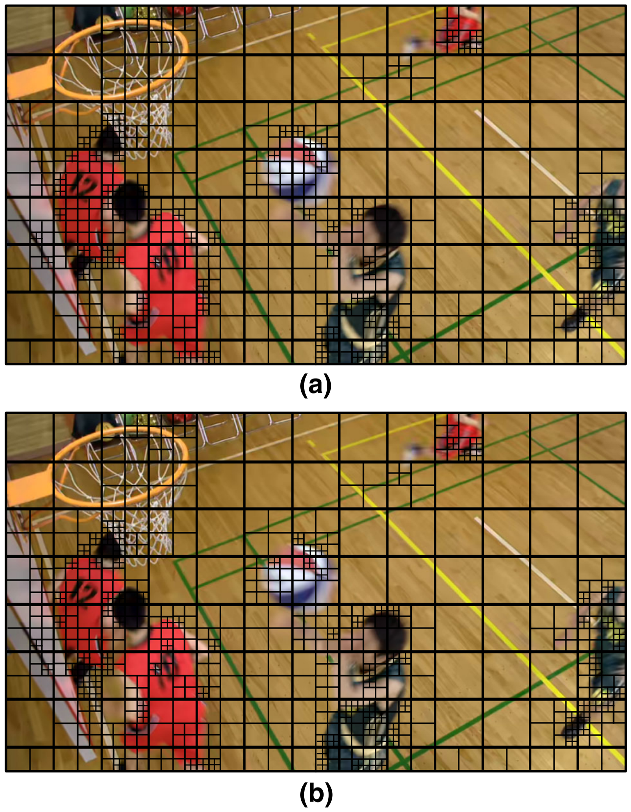


Fig. 7 CTUs divided into CUs in the 100th frame of the BasketballDrill video sequence encoded with QP 32 by **a** the original HM encoder and the **b** HM encoder with the proposed Coding Tree early termination

respectively. The rightmost column of Table 6 shows the ratio between BD-rate and CCR, which is mostly useful for comparisons with similar works that yield different CCR levels, as we show later in Sect. 5.3.

Figure 8 shows R-D efficiency results for the two encoder versions, i.e. the original (*HM12*) and the one with the proposed early termination method (*CT ET*). Figure 8(a) and (b) show results for the *BQTerrace* and the *SteamLocomotive* video sequences, which are those that presented the best and worst R-D efficiency results, respectively. Notice that the charts show overlapped curves even in the worst-case video sequence, which means that the R-D efficiency of both encoder versions is almost the same, even though the computational complexity is 37 % smaller in the proposed method.

5.2 Method robustness analysis

To evaluate the robustness of the method proposed in this work, a set of experiments that deliberately alter the obtained classification trees was performed. The goal of such analyses was to identify how much the results presented in Sect. 5.1 depend on the attribute thresholds acquired during the training process and how much these thresholds depend on the video sequences used to train the classification trees. Four modifications were performed directly on the implemented trees and the results in terms of BD-rate, BD-PSNR and CCR (with reference to the original HM encoder) were observed and compared to the encoder with the original classification trees proposed in Sect. 4. Table 7 presents the average results for each set of experiments, which are detailed in the next paragraphs. For reference, the results obtained with the original decision trees are reproduced in the first line of Table 7 (*Training Set A*).

In the first and second set of experiments (*Threshold* + *5* % and *Threshold* − *5* %), the thresholds tested in each level of the classification trees were increased or decreased in 5 %, respectively. These are important evaluations because small variations on the threshold values are expected if some training conditions change, such as the number of video sequences, the QP, the characteristics of each video (resolution, frame rate, texture), and the number of frames. By forcing the thresholds to +5 or −5 %, we can evaluate the impact in terms of compression efficiency and computational complexity if a slightly different set of trees was used. The results in Table 7 show that the compression efficiency and the CCR do not change significantly in comparison to the *Training Set A*.

In the third set of experiments (*Termination-5* %), the thresholds were modified in such a way that, considering a uniform distribution of values for a certain attribute, the chances of terminating the CU splitting process earlier were decreased in 5 % (or, in other words, the chances of continuing the splitting process were increased in 5 %). Thus, the threshold values were increased or decreased in 5 %, accordingly to the choice that their tests represent. For example, if testing a certain attribute leads to an early termination if its value is smaller than x, threshold x was increased in 5 %, so that the chances of early terminating the CU splitting process were decreased in 5 %. Similarly, if testing a certain attribute leads to an early termination if its value is larger than x, threshold x was decreased in 5 %, so that the chances of early terminating the CU splitting process were decreased in 5 %. Table 7 shows that the CCR decreased 7 % and the compression efficiency (BD-rate) increased 0.02 % in these experiments, respectively. This was already expected, since the chances of early terminating the CU splitting process were decreased in 5 % (thus, more optimal decisions are performed). These experiments may provide initial insight for the future development of a complexity control algorithm that can dynamically adjust the amount of early terminations according to a desired level of computational complexity reduction.

Finally, the fourth set of experiments (*Training Set B*) analyzed the influence of the video sequences used in the

Table 6 Rate-distortion-complexity results for the early termination

Video sequence	BD-rate (%)	BD-PSNR (dB)	CCR (%)	$\frac{\text{BD-rate}}{\text{CCR}}$
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

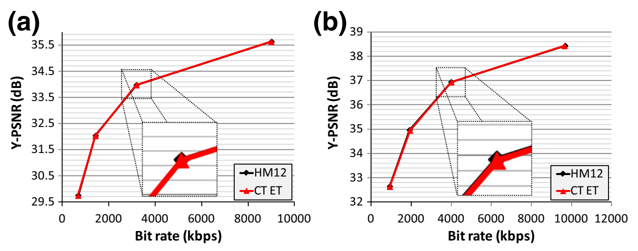


Fig. 8 Rate-Distortion efficiency for the **a** BQTerrace and **b** SteamLocomotive video sequences encoded with the original HM encoder and the HM encoder with the proposed early termination

Table 7 Robustness analysis results

Experiment	BD-rate (%)	BD-PSNR (dB)	CCR (%)
Training Set A	+0.28	−0.01	37
Threshold +5 %	+0.32	−0.01	35
Threshold −5 %	+0.21	−0.01	34
Termination −5 %	+0.26	−0.01	30
Training Set B	+0.37	−0.02	32

training process on the overall performance of the proposed method. For that purpose, the whole process described in Sect. 4 was repeated for the video sequences belonging to set *B* of Table 1 (instead of those in set *A*). The new obtained decision trees were then implemented in the HM encoder and evaluated with the video sequences belonging to set *A* of Table 1 (instead of those in set *B*). Notice in the last line of Table 7 that the CCR and the compression efficiency results are slightly worse, but still very similar to those obtained with training set *A* (first line), which means that the method presents good results regardless of the training/testing set.

Obviously, as the training video sequences used in the fourth set of experiments are not the same as those used in Sect. 4, the obtained classification trees are slightly different. However, their structure is very similar and the main differences are in the threshold values. Table 8 shows the topological characteristics and the decision accuracy of the trees trained with set *B*. Notice that the tree depth, the number of test nodes and the number of leaves are very close to those in the trees presented in Table 4. Similarly, the classifier accuracy and the percentage of incorrect coding tree depth choice are close to those presented in Table 5, even though they are slightly smaller. As expected, this small decrease in accuracy leads to the slightly smaller compression efficiency results presented in the last line of Table 7.

5.3 Comparison with related works

The best related works found in the literature have been analyzed and compared to the schemes proposed in this article. In order to allow for a fair comparison, only those works that reported BD-rate results and CCR values relative to the overall encoding process using the original HM as reference were selected. Besides, all the compared works were also tested with the *Main* profile, the RA configuration, QPs 22, 27, 32, 37, and at least seven video sequences with four different spatial resolutions.

Table 9 presents the compared results in terms of BD-rate, CCR and the ratio between these two values. We can conclude from the table that the early termination scheme proposed in this article achieved the best BD-rate/CCR relationship among all related works. 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. Also, notice that the best related work [9] is actually only applicable to intra-predicted CUs, which means that its complexity reductions are only achieved in *All Intra* (AI) temporal configurations.

Finally, it should be noticed here that this work presents a detailed description on top of the research published in

Table 8 Topological characteristics and accuracy of classification trees obtained with Training Set B

CU size	Depth	Test nodes	Leaves	Classifier accuracy (%)	Incorrect depth (%)
64 × 64	7	9	22	82.9	8.5
32 × 32	8	20	33	83.6	9.0
16 × 16	9	26	45	84.6	8.1

Table 9 Comparison with related works

Related work	BD-rate (%)	CCR (%)	$\frac{\text{BD-rate}}{\text{CCR}}$
Seunghyun [9]	+0.6	50	1.20
Correa [10]	+0.8	40	2.00
Jong-Hyeok [11]	+1.2	48	2.50
Goswami [12]	+1.7	38	4.47
Jian [13]	+1.9	43	4.42
Shen [16]	+1.4	45	3.11
Proposed	+0.28	37	0.76

[25], mainly focusing on the coding tree structure decision and on the method robustness analysis.

6 Conclusions and future work

This article 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 computed during normal HEVC encoding. Therefore, a similar strategy can be seamlessly incorporated to any other HEVC encoder implementation with very small increase in the computational burden of the encoding.

This work also presented a set of four extensive experiments that were performed to assess the robustness of the proposed method. The experiments revealed that the high accuracy of the classification trees and the high R-D efficiency of the modified HEVC encoder are not biased to the video sequences used for training and testing, so that other different sequences still yield very similar results. Besides, the robustness analysis provided initial insightful conclusions for the future development of a complexity control mechanism that is able to adjust the classification tree thresholds dynamically according to the desired level of computational complexity reduction.

Acknowledgments The authors would like to acknowledge the valuable help of Giovanni Ávila, Eliézer Ribeiro, Douglas Corrêa and Iago Storch, who conducted the method robustness analysis presented in Sect. 5.2. The authors also acknowledge financial aid from FAPERGS-Brazil, CAPES-Brazil, CNPq-Brazil, and FCT-Portugal. This work was supported by FAPERGS-Brazil, CAPES-Brazil, CNPq-Brazil, the project FCT-CAPES (4.4.1.00 CAPES) (FCT/1909/27/2/2014/S), the R&D Unit UID/EEA/50008/2013 and the FCT grant SFRH/BSAB/113682/2015.

References

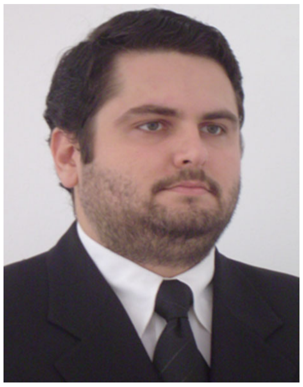
1. Vanne, J., Viitanen, M., Hamalainen, T., & Hallapuro, A. (2012). Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs. *IEEE Transactions on Circuits and Systems for Video Technology, 22*, 1885–1898.
2. Correa, G., Assuncao, P., Agostini, L., & da Silva Cruz, L. A. (2012). Performance and computational complexity assessment of high efficiency video encoders. *IEEE Transactions on Circuits and Systems for Video Technology, 22*, 1899–1909.
3. Sanchez, G., Corrêa, M., Noble, D., Porto, M., Bampi, S., & Agostini, L. (2012). Hardware design focusing in the tradeoff cost versus quality for the H.264/AVC fractional motion estimation targeting high definition videos. *Analog Integrated Circuits and Signal Processing, 73*, 931–944.
4. Sanchez, G., Zatt, B., Porto, M., & Agostini, L. (2015). Hardware-friendly HEVC motion estimation: new algorithms and efficient VLSI designs targeting high definition videos. *Analog Integrated Circuits and Signal Processing, 82*, 135–146.
5. Kuhn, P. M., & Kuhn Peter, M. (1999). *Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation*. Dordrecht: Kluwer Academic Publishers.
6. Tourapis, A. M. (2002). Enhanced predictive zonal search for single and multiple frame motion estimation. *Electronic Imaging, 2002*, 1069–1079.
7. Bossen, F., Bross, B., Suhring, K., & Flynn, D. (2012). HEVC complexity and implementation analysis. *IEEE Transactions on Circuits and Systems for Video Technology, 22*, 1685–1696.
8. ISO/IEC-JCT1/SC29/WG11 (2013). High efficiency video coding (HEVC) test model 12 (HM 12) encoder description. Vienna, Austria.
9. Seunghyun, C., & Munchurl, K. (2013). Fast CU splitting and pruning for suboptimal CU partitioning in HEVC Intra coding. *IEEE Transactions on Circuits and Systems for Video Technology, 23*, 1555–1564.
10. Correa, G., Assuncao, P., Agostini, L., & Silva Cruz, L. (2014). Complexity scalability for real-time HEVC encoders. Journal of Real-Time Image Processing, 1–16.
11. Jong-Hyeok, L., Chan-Seob, P.,& Byung-Gyu, K. (2012). Fast coding algorithm based on adaptive coding depth range selection

for HEVC. In *IEEE international conference on consumer electronics-Berlin* (*ICCE-Berlin*) (pp. 31–33).

12. Goswami, K., Kim, B.-G., Jun, D.-S., Jung, S.-H., & Choi, J. S. (2014).Early coding unit (cu) splitting termination algorithm for high efficiency video coding (HEVC). ETRI Journal.
13. Jian, X., Hongliang, L., Qingbo, W., & Fanman, M. (2014). A fast HEVC inter CU selection method based on pyramid motion divergence. *IEEE Transactions on Multimedia, 16*, 559–564.
14. Lampert, C. H. (2006). Machine learning for video compression: Macroblock mode decision. In *18th international conference on pattern recognition, ICPR* (pp. 936–940).
15. Garcia, R., Ruiz Coll, D., Kalva, H., & Fernandez-Escribano, G. (2013). 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.
16. Shen, X., & Yu, L. (2013). CU splitting early termination based on weighted SVM. *EURASIP Journal on Image and Video Processing, 2013*, 4.
17. ISO/IEC-JCT1/SC29/WG11 (2013). High Efficiency Video Coding (HEVC) text specification draft (vol. 10). Geneva, Switzerland.
18. Correa, G., Assuncao, P., Agostini, L., & da Silva Cruz, L. A. (2011). Complexity control of high efficiency video encoders for power-constrained devices. *IEEE Transactions on Consumer Electronics, 57*, 1866–1874.
19. ISO/IEC-JCT1/SC29/WG11 (2012). Common test conditions and software reference configurations. Geneva, Switzerland.
20. Bjontegaard, G. (2001). Calculation of average PSNR differences between RD-curves. Austin, Texas.
21. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explor. Newsl., 11*, 10–18.
22. Quinlan, J. R. (1983). C4.5: Programs for machine learning. Burlington: Morgan Kaufmann Publishers.
23. Kullback S. & Leibler R. A. (1951) On information and sufficiency (pp. 79–86).
24. Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 international conference on artificial intelligence* (*IC-{AI}'2000*) (pp. 111–117).
25. Correa, G., Assuncao, P., Agostini, L., & da Silva Cruz, L. (2015). Fast HEVC encoding decisions using data mining. *IEEE Transactions on Circuits and Systems for Video Technology, 25*(4), 660–673.

Guilherme Correa received the B.Sc. degree in Computer Science from the Federal University of Pelotas (UFPel), Brazil, in 2009, the M.Sc. degree in Computer Science from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2010, and the Ph.D. degree in Electrical and Computer Engineering from the University of Coimbra (UC), Portugal, in 2015. Now he is a postdoctoral researcher with the Group of Architectures and Integrated Circuits (GACI) at UFPel, where he works on video coding algorithms, complexity-aware video coding and digital systems design.

Pedro Assuncao received the Licenciado and M.Sc. degrees in Electrical Engineering from the University of Coimbra, Portugal, in 1988 and 1993, respectively, and the Ph.D. in Electronic Systems Engineering from the University of Essex, UK, in 1998. He is currently Professor of Electrical Engineering and Multimedia Communication Systems at the Polytechnic Institute of Leiria and researcher at the Institute for Telecommunications, Portugal. He is author/co-author of more than one hundred papers in conferences and journals, seven book chapters and four US patents. His current research interests include 2D, 3D and plenoptic video coding, complexity control and networking, multiple description coding, error concealment and quality evaluation.



Luciano Agostini received the B.Sc. degree in Computer Science from the Federal University of Pelotas (UFPel), Brazil, in 1998 and the M.Sc. and Ph.D. degrees from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2002 and 2007, respectively. He is a Professor since 2002 at UFPel. Since 2013 he is the UFPel Executive Vice President for Research and Graduation Studies. He is also the leader of the Group of Architectures and Integrated Circuits (GACI) at the same university. He has more than 100 published papers in journals and conference proceedings. His research interests include 2D and 3D video coding, algorithmic optimization, arithmetic circuits, FPGA based design and microelectronics. Dr. Agostini is a Senior Member of IEEE and he is a member of the IEEE CES, CAS, and SPS. He is also member of ACM, SBC, SBMicro.

Luis A. da Silva Cruz received the Licenciado and M.Sc. degrees in Electrical Engineering from the University of Coimbra, Portugal, in 1989 and 1993, a M.Sc. degree in Mathematics and a Ph.D. degree in Electrical Computer and Systems Engineering from Rensselaer Polytechnic Institute (RPI), Troy, NY, US in 1997 and 2000 respectively. He has been with the Department of Electrical and Computer Engineering of the University of Coimbra in Portugal since 1990 first as a Teaching Assistant and as an Assistant Professor since 2000. He is a researcher of the Institute for Telecommunications, Portugal, where he works on video processing and coding, wireless communications and medical image processing.