Complexity Control of High Efficiency Video Encoders for Power-Constrained Devices

Guilherme Corrêa, *Student Member*, IEEE, Pedro Assuncao, *Member*, IEEE,
Luciano Agostini, *Senior Member*, IEEE, and Luis A. da Silva Cruz, *Member*, IEEE

Abstract — *The emerging High Efficiency Video Coding (HEVC) standard is expected to require much more processing power than its predecessors due to the higher algorithmic complexity of new coding tools and associated data structures. This paper proposes a novel complexity control method for the near future HEVC encoders running on power-constrained devices. The proposed method is based on a decision algorithm that dynamically adjusts the depth of the Coding Units (CU) defined by quad-tree structures. New evidence about the relationship between CU depth and coding complexity is used to selectively constrain the CU depth in order to not exceed a predefined complexity target. The experimental results show that the encoder computational complexity can be downscaled by up to 60% at the cost of negligible loss of rate-distortion (RD) performance. The proposed method finds application in the near future multimedia portable devices using HEVC codecs*[1].

Index Terms — Complexity control, complexity scalability, High Efficiency Video Coding (HEVC), Coding Unit (CU).

I. INTRODUCTION

In the last decades, the rapid advances of semiconductor technologies fostered a large expansion in the consumer market of multimedia-ready devices due to the continuous increase of computational resources and availability of reliable communication infrastructures. Nowadays, digital TVs, portable computers, PDAs and even cell phones are among the most popular consumer equipments able to receive and display high-resolution video in real time. Very common are also those devices which can capture and send digital video through wired and wireless networks. Furthermore, the current trend in most portable devices with embedded digital cameras is to include the capability of encoding and decoding high-resolution digital video streams.

Despite recent evolution in portable devices, particularly in communications technology and computational power, the limited battery capacity still imposes major constraints in multimedia applications demanding for high computational power, such as those dealing with video coding/decoding.

In such applications, the user quality of experience is limited by the reduced battery capacity. While academy and industry keep researching and developing new and more efficient solutions for increasing the battery life, design engineers try to decrease power consumption and use computational resources wisely in their systems, through the application of low-power design techniques in hardware architectures and software algorithms.

Previous studies which examine typical use scenarios of portable devices have shown that a very significant amount of power consumption (from 40% to 60%) is related to video encoding and decoding, with the encoder typically requiring the largest share of computing time and power consumption [1]-[4]. It is claimed that more than two-thirds of this computational complexity corresponds to the encoding process, whereas the rest is divided between transmission and I/O operations [5]. Even though this is just an estimate for low-resolution video, the adoption of higher resolutions will certainly increase even more the computational complexity required for video coding, since greater computational effort will be necessary to process the increasingly larger amounts of video information. Furthermore, a consequence of current video coding standard evolution is that more efficient signal processing tools are significantly augmenting the number of operations-per-pixel required in the near future high efficiency video codecs. This is the case of the future video coding standard, High Efficiency Video Coding (HEVC), expected to be finished by 2013 [6]. The aim of HEVC is to achieve 50% of bit rate reduction in comparison with H.264/AVC High Profile [7], at the same subjective image quality. However, to reach such goal, the computational complexity should not exceed that of H.264/AVC High Profile by more than a factor of three [8]. Even though such coding efficiency goal and complexity constraint are taken into consideration during the development of HEVC, there is as yet no published research proposing methods to deal with the problem of complexity control or complexity scalability for this emerging standard. Although the standard is still under development and aggregating new coding functions, the current version of the HEVC Test Model (HM) reference software [9] already includes coding tools which are much more complex than those of H.264/AVC.

In the past, as cited in [2], several works addressing complexity-aware video coding have been proposed to deal with energy constraints in portable devices but none of them applies to the particular case of the computational complexity

[1] G. Corrêa, and L. A. da Silva Cruz are with the Department of Electrical and Computer Engineering (Faculty of Sciences and Technology, University of Coimbra) and with the Instituto de Telecomunicações (IT), Polo II – Universidade de Coimbra, Pinhal de Marrocos, 3030-290, Coimbra, Portugal (e-mails: guilherme.correa@co.it.pt, luis.cruz@co.it.pt).

P. Assunção is with the Polytechnic Institute of Leiria and with the Instituto de Telecomunicações (IT), Morro do Lena - Alto do Vieiro, 2411-901, Leiria, Portugal (e-mail: amado@co.it.pt).

L. Agostini is with the Group of Architectures and Integrated Circuits, Federal University of Pelotas, Pelotas-RS, 96010-900, Brazil (e-mail: agostini@inf.ufpel.edu.br).

of HEVC, which is highly dependent on its inherent data structures. In the current HEVC Test Model (HM), Coding Unit (CU) trees are new data structures, which require a set up process demanding most of the encoding computational complexity (see sections II and IV).

This paper proposes a method for controlling the computational complexity required to define CU tree structures by limiting the number of coding decision tests and comparisons. A novel feature of the proposed method introduces complexity scalability into HEVC, such that the encoder computational complexity can be downscaled by up to 60% with negligible loss of rate-distortion (RD) performance.

The paper is organized as follows: Section II summarizes the most relevant HEVC tools and explains the operation of the CU tree-based coding. Section III briefly reviews important works on complexity-aware video coding. Section IV presents an analysis of the CU tree structure computational complexity and the basics of the proposed method. Section V details the method for complexity control proposed in this paper. Section VI presents experimental results and comparisons. Finally, section VII concludes the paper.

II. HIGH EFFICIENCY VIDEO CODING

In the current test model of HEVC there are six possible configurations for normalized experiments to be executed over the reference software, called Tool Experiments [10]. These configurations are combinations of two possible complexity/efficiency settings, *High Efficiency* (HE) or *Low Complexity* (LC), and three coding/access settings, *Intra Only*, *Random Access* or *Low Delay*. The HE setting is used for high coding efficiency, whereas the LC setting defines a coding configuration with low computational complexity (which does not provide the best coding efficiency). *Intra Only* mode uses no temporal prediction, *Random Access* allows the use of future frames as reference in temporal prediction and *Low Delay* uses only previous frames as reference. The resulting six combinations are the following [10]: (1) Intra Only, High Efficiency; (2) Intra Only, Low Complexity; (3) Random Access, High Efficiency; (4) Random Access, Low Complexity; (5) Low Delay, High Efficiency; (6) Low Delay, Low Complexity. Since this work is focused on computational complexity control of video coding operations carried out in devices with power constraints, the *Low Delay*, *Low Complexity* configuration was used.

As pointed out before, CU tree structures are responsible for most of the encoder computational complexity. Even though CUs only represent a group of pixels to be encoded together as a single unit, the definition of its size is a very complex task, because it involves encoding the image using all possible CU sizes allowed by HEVC, comparing their RD cost and finally choosing the best one. All current video coding standards use sub-divisions of frames into groups of pixels to be encoded as single units. In H.264/AVC, the general sub-division is a macroblock (MB) of 16x16 pixels, which may be further partitioned into 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4 blocks [7]. In HEVC CUs are much more flexible coding structures allowing a wider range of block sizes. CU sizes larger than 16x16 pixels are especially useful at high resolutions because larger images contain larger homogeneous areas that can be encoded in larger blocks, which can be more efficiently compressed by transforms and intra-prediction.

The tree-based structure was added to the first test model of HEVC and in the current version of HM [9] each CU is allowed to have 64x64, 32x32, 16x16, or 8x8 pixels. The encoder decides the final tree structure based on the Rate-Distortion Optimization (RDO) technique, where all the possible partitioning configurations definable in a quad-tree are tested and compared [11].

Fig. 1 shows an example of a 128x128 pixels area divided into variable-sized CUs. The first 64x64 area (I) is divided into four 32x32 CUs (second depth). The second 64x64 area (II) is encoded as just one 64x64 CU (first depth). The third 64x64 area (III) is divided into four 32x32 CUs, two of which are divided into four 16x16 CUs, one of which is divided into four 8x8 CUs (fourth depth). Finally, the fourth 64x64 area (IV) is divided into four 32x32 CUs and two of them are divided into four 16x16 CUs (third depth).

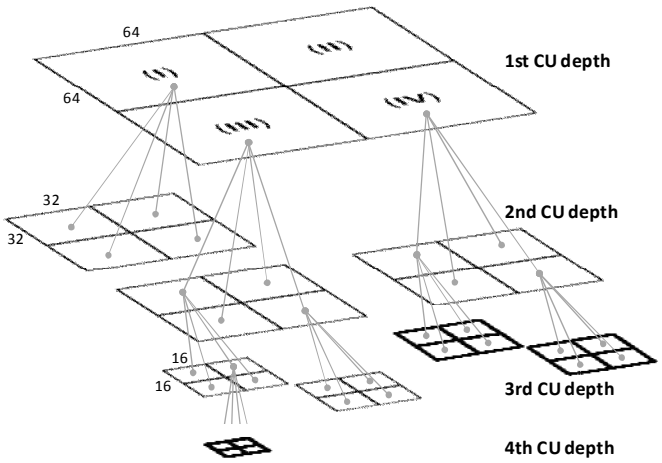

Fig. 1. Example of quad-tree-based CU division into smaller CUs.

In the RDO process implemented in HM, each frame is initially divided into CUs of 64x64 pixels, which are further divided into smaller CUs, according to the maximum split depth allowed. These initial CUs are defined at the first depth level (CUDepth = 1) and each one is split in four identical smaller CUs of 32x32 pixels. These four new CUs are said to belong to the second CU depth (CUDepth = 2) and each one is divided into four smaller CUs of 16x16 pixels. The new CUs are now at the third depth level (CUDepth = 3) and can be divided still once more into four 8x8 CUs, which are at the fourth and last CU depth level (CUDepth = 4). The maximum split depth defines the smallest block size allowed for coding.

When the splitting process is finished, each leaf CU is encoded according to the configuration used (HE or LC). The RD cost of encoding the CU at this level is stored and afterwards summed with its three sibling CUs costs. This sum

is compared to the RD cost of encoding the four CUs as just one at the immediately previous depth (e.g., comparing the RD costs of encoding four 8x8 CUs with the RD cost of encoding them as just one 16x16 CU). If the cost is indeed smaller in the first case, then these four CUs shall be separately encoded. Otherwise, the four CUs are merged and the process is repeated for the immediately previous depth (e.g., comparing the costs of encoding four 16x16 CUs with the cost of encoding them as just one 32x32 CU). This process repeats until the 64x64 CU (CUDepth = 1) is reached.

Even though this encoding method returns the best possible tree structure, its computational complexity is very high. In order to obtain the RD costs used in the comparisons described above, the encoder must try using all valid coding modes to encode each possible CU, which includes the computation of intra-prediction, motion estimation and compensation, transform and quantization, entropy coding and deblocking operations. Besides, for each possible CU, a Prediction Unit (PU) shape and a Transform Unit (TU) tree structure must be defined, which means that this complex RDO process still has further embedded RDO processes to define other nested structures. Heuristics or good scene characteristics estimation methods are thus necessary to predict the possible behavior of future frames concerning the optimal depth to be used when coding each CU.

III. Related Work

Several methods have been proposed in the last few years with the objective of controlling the computational complexity of video coding and decoding algorithms with the ultimate goal of reducing energy consumption and expanding the autonomy of portable devices [1]-[2]. Many of these previous works are focused on the motion estimation (ME) and coding mode decision (MD) processes, since these are the most complex functions, also accounting for the largest processing time shares in current video encoders [12]-[18]. In these methods, complexity control is accomplished by performing ME at different levels of precision and complexity. Higher precision levels return the best results but require a larger amount of calculations, thus spending more energy.

In [14], motion vector fields are processed in three stages with ascending complexity levels, performing edge detection to minimize the number of pixel comparisons per MB by limiting these operations to the areas around the detected edges. In [15], [17], a similar complexity control algorithm is proposed to dynamically adjust the ME search method, either adopting a fast full search approach or a combination of different fast search techniques. In the hardware architecture proposed in [17] it is worthwhile to point out that the complexity budget is measured in clock cycles and an adaptive threshold is used to adjust the budget per frame. A similar approach is used in this paper.

In [16], the computational complexity is managed through adjustments in parameters that jointly control both ME and MD operations, such as the search area in ME and the number of candidate coding modes in MD. A rather different technique is proposed in [18], where partial information, such as image format and spatio-temporal activity, is extracted from the encoder input signal to configure ME parameters dynamically.

A method based on controlling the MD computational complexity through an estimation of the RD cost statistics followed by optimal coding mode decision for each MB is described in [19]. The set of possible coding modes is either increased or decreased according to time requirements of any specific application. A different work is presented in [20] based on adaptive early-termination of the MD process by predicting the most probable modes, based on histograms from neighboring MBs.

Besides ME and MD, the computational complexity of other encoder modules can also be controlled through some parametric adjustment, such as proposed in [21], where eleven parameters are used for that purpose. Twenty possible complexity operating points are defined by different combinations of parameter values. Also in [22], the complexity of entropy coding, deblocking filter, ME and MD operations are controlled through the use of four different parameters, which adjust the accuracy/complexity of each module operation separately.

Other research works modify the RDO process in order to include computational complexity in the cost function [23]. Then, the joint rate-distortion-complexity (RDC) cost is used in the coding mode decision process. In [24], the RDO process is also modified to consider complexity by eliminating either the bit rate or the distortion term in the RD cost calculation through a simple conversion of its value, which is summed to the remaining term. In [25], [26], a set of parameters is used to control the computational complexity of encoding modules. For each complexity level, an analysis of the relationship between the energy consumption, bit rate and image quality is performed in order to choose the best values for the complexity control parameters. In [27], the encoder is divided into three parts controlled by parameters which define the target computational complexity. The control is performed through joint minimization of energy consumption and image coding distortion.

All these works have been proposed and implemented for video coding algorithms which precede HEVC. Most of them control the computational complexity associated with ME and MD because these are the most time consuming functions in previous coding algorithms. As already pointed out, HEVC is based on the CU tree structure, and to the best of the authors' knowledge no previous work addresses complexity control of video encoding using the specific coding features associated with this new data structure. Next section describes the basics of the complexity control scheme proposed in this paper.

IV. Coding Complexity of CU Tree Structures

The computational complexity of the high efficiency video encoder HM was evaluated for each different coding tool (e.g. intra and inter prediction and adaptive loop filter) as the corresponding processing time. In this work, this information

was obtained through the use of a software profiler which reports processing times without considering other operating system functions. The profiler was executed on the HM software compiled for a 64-bit processor architecture and running on a 2.27 GHz processor.

The computational complexity of each coding tool used in HEVC and its dependence on the CU tree depth was experimentally determined using three 1280x768 pixel video sequences, each one composed of 504 frames: *Basketball* (few details and medium amount of colors per frame, high motion), *BQTerrace* (medium amount of details and few colors per frame, medium motion), and *Cactus* (large amount of details and colors per frame, high motion). The total processing time was found to be very similar for the three video sequences.

Fig. 2 shows the average percentage of computational complexity demanded for encoding CUs belonging to each depth of the tree structure. CU depths are presented in the y-axis of Fig. 2, and the computational complexity is presented in the x-axis. As mentioned before the nature of the data structures used in HEVC leads to nested encoding loops, such that CUs at higher tree depths are encoded inside CUs at smaller tree depths. This is shown in Fig. 2 as follows: for each CU depth (vertical axis), the computational complexity is divided in three components: (1) the complexity of performing inter prediction for CUs at the current depth (in grey); (2) the complexity of coding the same image area as CUs at a higher depth (in white); and (3) the complexity of other operations (in black).

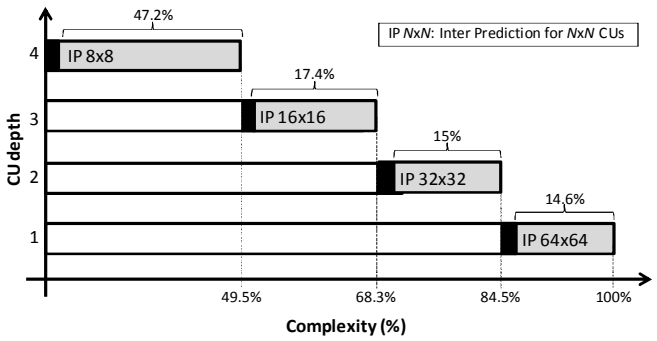

Fig. 2. Computational complexity of encoding CUs in each tree depth.

Due to this nested encoding structure, the percentage of computational complexity demanded to encode CUs at the first tree depth is 100% (bottom bar in Fig. 2), which means that the maximum overall complexity corresponds to the largest CU. This is because it includes the complexity of coding CUs at the second tree depth (in white), in addition to the complexity of inter prediction for 64x64 CUs (*IP 64x64*, in grey) and also the complexity of other small operations (in black). Similarly, the complexity of compressing CUs in the second tree depth (CU depth = 2 in Fig. 2) includes the complexity of coding CUs at the third tree depth (in white), the complexity of inter prediction for 32x32 CUs (*IP 32x32*, in grey) and the complexity of other various operations (in black). The same structure applies to the third depth. Finally, for CU depth = 4, the complexity includes only the complexity of inter prediction for 8x8 CUs (*IP 8x8*, in grey) and the complexity of other small operations (in black), since no further tree depths are allowed.

Notice that, even though the smallest CU depths present the largest total percentages of computational complexity in Fig. 2, the highest CU depths are the actual responsible for most of the encoding computational complexity (*IP 8x8*). For example, although compressing CUs in the third depth (composed by 16x16 CUs) represents 68.3% of the overall computational complexity, in fact only 17.4% is the real complexity associated to this CU size, whereas almost all the remaining complexity is dedicated to inter prediction for 8x8 CUs (*IP 8x8*) and other operations (49.5%). These results provide relevant insight about the distribution of complexity over CUs of different sizes and take into account the nested coding structure used in HEVC. An important conclusion is that the major contributors to the overall computational complexity are the inner CUs, which are much more numerous.

The behavior of CU coding along the temporal domain was also investigated. Based on the concept of stationarity (also called *stillness*), defined as the tendency of a video sequence to comprise large image areas with either no or low motion between frames, an experimental study was carried out in order to characterize CU tree depth variations in co-located areas of neighboring frames. Fig. 3 represents the maximum CU tree depth used in a random co-located 64x64 area for all frames of the *BQTerrace* video sequence. The figure shows that the maximum CU tree depth is kept constant during long periods of the sequence. It means that once a CU tree depth is used in a determined area of the video, the same depth tends to be used for a long period in co-located areas of adjacent frames before it changes to a different value.

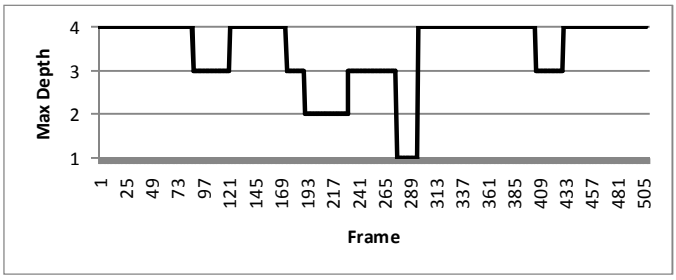

Fig. 3. Maximum CU tree depth of a random co-localized 64x64 area for the *BQTerrace* video sequence.

V. PROPOSED METHOD FOR COMPLEXITY CONTROL

The basis for the complexity control method introduced in this work was the relevant evidence presented in the previous section. The aim is to constrain the maximum number of CUs tested in the RDO process in order to avoid processing CUs at large tree depths when this is found to have high complexity cost and bring small encoding gain. This restriction allows the video sequence to be encoded without exceeding a target complexity defined by the user or by the device itself (e.g., based on the remaining battery life). To achieve this, from time to time the algorithm encodes some frames without constraining the tree depths used in the CUs. The largest

depths used in this unconstrained encoding process are then used as maximum depths allowed in the next frames.

During the encoding process, the proposed method defines two types of frames: the unconstrained frames (*Fu*) and the constrained ones (*Fc*), defined according to their associated computational complexity. Each *Fu* frame is followed by *Nc* constrained frames of type *Fc* (Fig. 4).

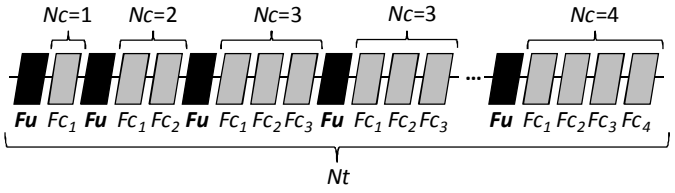

Fig. 4. Operation of the complexity control algorithm. *Fu*: unconstrained frames; *Fc:* complexity controlled (constrained) frames.

Unconstrained frames *Fu* are encoded after encoding *Nc* consecutive *Fc* frames and then the control algorithm defines *Nc* for the next group of frames as a function of the available complexity.

When an unconstrained frame *Fu* is being encoded, the maximum tree depth used in each 64x64 area (computed by the RDO process) is stored into a vector of size *n* (here called *MaxCUD_history[n]*), where *n* is the number of 64x64 areas in the frame. For example, if Fig. 1 represented a frame of 128x128 pixels, the vector would contain *n*=4 positions and would store the values 2, 1, 4, and 3, which correspond to the maximum depths used in areas I, II, III, and IV, respectively.

When the constrained frames *Fc* are being encoded, the control algorithm does not allow the RDO process to test all the possible quad-tree configurations in its optimization process. Instead, the algorithm uses the information in *MaxCUD_history[n]* and limits the maximum tree depth tested in each 64x64 area to the value saved in the corresponding vector position. Since co-located areas in neighbor frames tend to have similar maximum depths, it is expected that coding efficiency is not harmed by limiting the RDO process to not exceed the tree depths values used in previous frames in the same image regions.

A target complexity is defined as a percentage of the maximum possible complexity that can be used to encode a predefined temporal segment of the video sequence. This is given by the maximum possible processing time that can be used to encode in real time all frames of the temporal segment. In portable devices, the target computational complexity can be either derived from a user-defined parameter or directly computed according to the battery energy level. The calculation of *Nc* is based on this target computational complexity *Tt*, on the complexity already spent in the encoding process *Td*, and on a prediction of the encoding complexity for the remaining frames within the temporal segment of the video sequence *Tp*.

When encoding a *Fc* frame, if the previously predicted complexity *Tp* plus the complexity already spent in previous frames is smaller than the target *Tt*, then more computation effort can be allocated and less frames need to be constrained to the maximum depths of the last unconstrained frame. Thus, *Nc* can be decreased. Otherwise, *Nc* is increased. A new value for *Tp* is calculated after coding each group of *Nc*+1 frames according to (1)

$$Tp = \frac{Te \cdot (Nt - Nd)}{(Nc + 1)}, \tag{1}$$

where *Te* is the encoding complexity of the last *Fu* frame plus the encoding complexity of the last *Nc* frames of type *Fc*, *Nt* and *Nd* are the total number of frames in the video temporal segment and the number of frames processed so far, respectively. The pseudo-code of the computational complexity control algorithm proposed in this paper is detailed in Fig. 5.

```
1   Nt ← total number of frames in the video
2   Tt ← target complexity to encode the video
3   FR ← frame rate
4   Td ← 0
5   Nd ← 0
6   Nc ← 0
7   n ← (frame_width · frame_height) ÷ (64 · 64)
8   for i  from 0 to n − 1 do
9     encode CUi
10    MaxCUD_history[i] ← maximum depth used in CUi
11  Nd++
12  for i  from 0 to Nc − 1 do
13    for j  from 0 to n − 1 do
14      if Nc = FR
15        encode CUj with maximum depth
16                  MaxCUD_history[j] − 1
17      Else
18        encode CUj with maximum depth
19                  MaxCUD_history[j]
20    Nd++
21  Te ← complexity from lines 8 to 20
22  Td ← Td + Te
23  Tp ← result of equation (1)
24  if ((Td+Tp > Tt · 1.1 ) AND (Nc < FR))
25    Nc++
26  else if ((Td+Tp < Tt · 0.9) AND (Nc > 0))
27    Nc--
28  if Nd < Nt
29    go to line 8
```

Fig. 5. Pseudo-code of the complexity control algorithm.

The algorithm includes a coding test (lines 14 to 16) which allows decreasing by one unit the values stored in the *MaxCUD_history[n]* vector. This complexity control function is enabled when *Nc* has already been increased a large number of times but the predicted encoding complexity is still higher than the target. This way, a larger complexity decrease is enforced by using a smaller depth and thus reducing the maximum CU depth. *Nc* can assume values in the range from 0 to *FR*, which is the frame rate given as frames per second (see lines 3, 24 and 26 in Fig. 5). The constraint *Nc* < *FR* ensures that the interval between two *Fu* frames is never larger than one second, which guarantees that the encoding RD performance is not increasingly reduced by the complexity control method. The algorithm also defines a tolerance of 10% when increasing or decreasing *Nc* (lines 24 and 26), as the actual complexity obtained after encoding and the previous forecast complexities will hardly be exactly equal. However, the deviation is rather small as described in the next section.

VI. EXPERIMENTAL RESULTS

The performance of this proposed method was evaluated by measuring the accuracy of complexity control under specific complexity targets and its influence on the RD efficiency of HM 2.0 encoder. As described in section IV, a specific software profiler was used to accurately measure the processing time, which is the metric used for computational complexity in this research. Since the experimental setup does not operate in real time, the total computational complexity available for encoding (i.e., 100%) was determined beforehand by encoding each sequence without imposing any complexity constraint. The average result was defined as the maximum available complexity available for encoding a sequence. A set of target complexities expressed as percentages of total available complexity was then defined to evaluate the performance of the proposed method. Table I shows the target complexities (i.e., encoding time) expressed as a percentage of the total complexity.

TABLE I
TARGET COMPLEXITIES

Target Complexity (%)	Encoding Time (s)
100	17,962
80	14,369
60	10,777
40	7,184

A. Complexity Control Behavior

The behavior of the complexity constraining algorithm was first analyzed. Fig. 6 shows the relationship between the target complexities employed in the encoding process and the actual complexities obtained for coding the three different video sequences. This figure shows that the proposed algorithm is quite accurate, since it is capable of controlling the complexity without major deviations from the target. Only one sequence (*BQTerrace*) presents a small deviation from the target complexity (roughly 7%). Fig. 7 presents the evolution of the number of constrained frames (i.e., *Nc*) during the encoding process for the four target complexities. The chart presents results only for the *Basketball* sequence since the other sequences exhibit a very similar behavior. The traces in Fig. 7 show that the algorithm is capable of controlling the number of constrained frames according to the limits imposed by the target complexity. When the total computational complexity is available (i.e., 100%), *Nc* remains equal to zero during the whole encoding process. On the other hand, when tighter target complexities are specified, *Nc* is increased until the predicted complexity either fits into the target complexity or reaches its upper limit (i.e., the full frame rate, 50 fps). Once the predicted complexity fits into the target, the algorithm stops increasing *Nc*, which remains constant until the end of the process since no more variations are necessary. Note that the full frame rate is never reached in Fig. 7, which means that the target complexity is achieved by proper control of *Nc*.

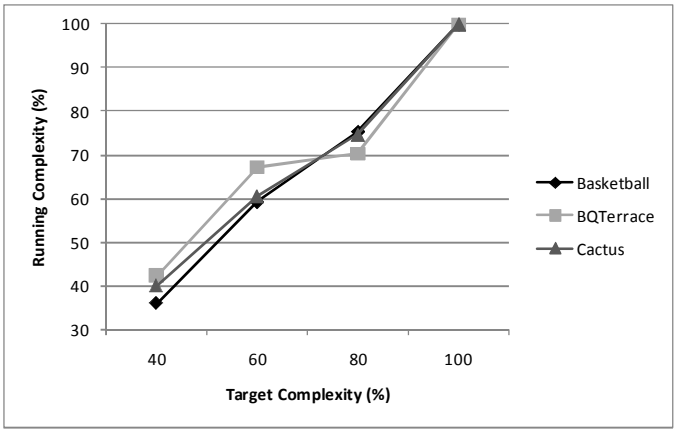

Fig. 6. Relationship between target complexity and actual complexity observed when encoding the three video sequences.

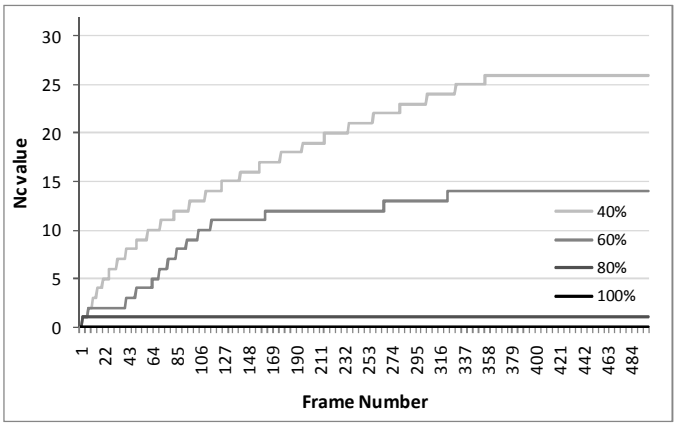

Fig. 7. Evolution of the number of constrained frames (*Nc*) according to each target complexity for the *Basketball* video sequence.

Figs. 8 to 10 show the effect of the complexity control algorithm on several coding performance indicators and variables. Only the results for 50 frames of *BQTerrace* coded at 60% of the target complexity are shown, since the data for all the other sequences and target complexities cases exhibit similar behavior. In Fig. 8 the black line presents the average CU tree depth per frame (vertical scale on the left side of the graph), whereas the grey line, (vertical scale on the right), presents the bit rate values obtained for each encoded frame. The average CU tree depths were computed according to (2), which is a weighted average of the CU depths with weights proportional to the corresponding CU area. In (2), *AvgDepth* is the weighted average of depths in the frame, $CUDepth_i$ is the depth at which the i_{th} CU is located, $CUArea_i$ is the area of the i_{th} CU measured in pixels, n is the number of CUs in the frame, and *TP* is the number of pixels in the frame.

$$AvgDepth = \frac{1}{TP}\sum_{i=1}^{n} CUDepth_i \cdot CUArea_i \qquad (2)$$

In Fig. 8, it is possible to notice that the bit rate is strongly dependent on the CU depths used to encode the frame. Whenever the depth is increased, the bit rate also increases, reflecting the fact that higher depths lead to use larger amounts of CUs, which are accompanied by more header information and, thus, higher bit rates. Some periodic bit rate

peaks appear in the graph of Fig. 8. The peaks are accompanied by the highest average CU depths and correspond to the unconstrained frames in which no complexity control was applied (*Fu* frames in the algorithm of section V). As these frames are not encoded with any computational restriction, they allow the use of CUs coded with high tree depths, increasing the bit rate.

Fig. 9 presents the average CU depths (in black) and the coded image quality per frame (in grey) quantified by the luminance PSNR. As in Fig. 9, some peaks appear in the unconstrained frames, which allow the use of all CU sizes in the whole frame. The PSNR varies less than 0.6 dB between the highest and the lowest point in the curve, which means that the method is able to maintain near constant image quality. The highest PSNR points are those which correspond to the unconstrained frames, as expected.

Finally, in Fig. 10, the average CU depth used in each frame (in black) is shown next to its encoding time in seconds (in grey). The peaks also appear in the unconstrained frames, as expected, but the points between two peaks present quite constant encoding time values, since they correspond to frames which were encoded with the same maximum CU depths. These results also show that the algorithm is efficiently distributing the available complexity between consecutive constrained frames.

B. Rate-Distortion-Complexity Performance

The performance of the computational complexity control was evaluated for rate, distortion and complexity. Fig. 11 shows the bit rate obtained after encoding the three video sequences at the four target computational complexities defined in Table I. The figure shows that the bit rate is maintained roughly constant from 100% to 60% of target complexity. When the target complexity is reduced to 40%, a small bit rate increase is observed: *Basketball* (+4.9%), *BQTerrace* (+2.8%) and *Cactus* (+5.7%). This is due to lower prediction performance in constrained areas, but the gain in complexity reduction is far more significant.

Fig. 12 shows the luminance PSNR results for the three video sequences, also encoded at the same target complexities. Similarly to Fig. 11, the PSNR is maintained practically constant from 100% to 60% target complexities. The differences are all under 0.1 dB when compared to the 100% target complexity. When the target complexity is reduced to 40%, a slightly larger PSNR drop is observed in the three video sequences: *Basketball* (0.77 dB), *BQTerrace* (0.14 dB) and *Cactus* (0.28 dB).

The rate-distortion efficiency as a function of the complexity is also presented in Fig. 13. As expected, the best results are obtained when no complexity control is applied (i.e., 100%). However, when complexity control is used, a small degradation in the encoding performance is observed, especially when tighter target complexities are defined. The traces in Fig. 13 correspond to the sequence *Basketball* and they represent the worst case observed in the experiments. The other sequences presented either similar or better results.

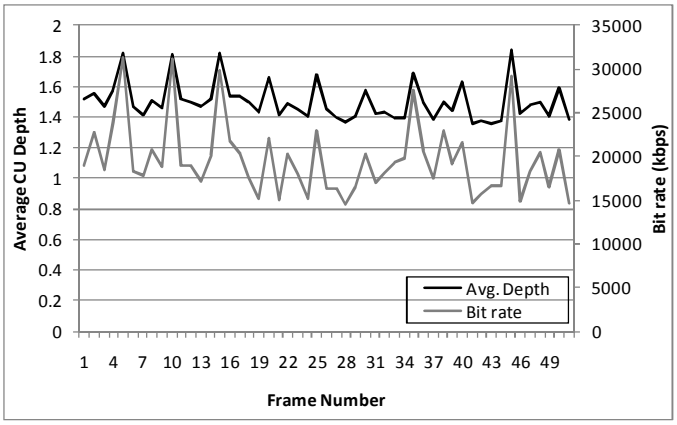

Fig. 8. Average CU depth and bit rate (kbps) per frame for the *BQTerrace* sequence, 60% target complexity.

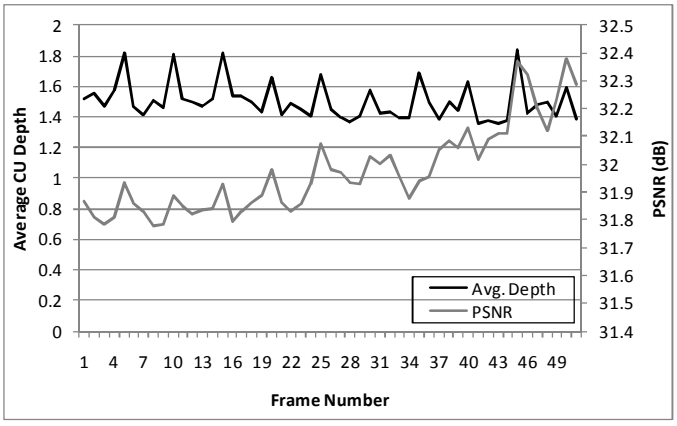

Fig. 9. Average CU depth and luminance PSNR (dB) per frame for the *BQTerrace* sequence, 60% target complexity.

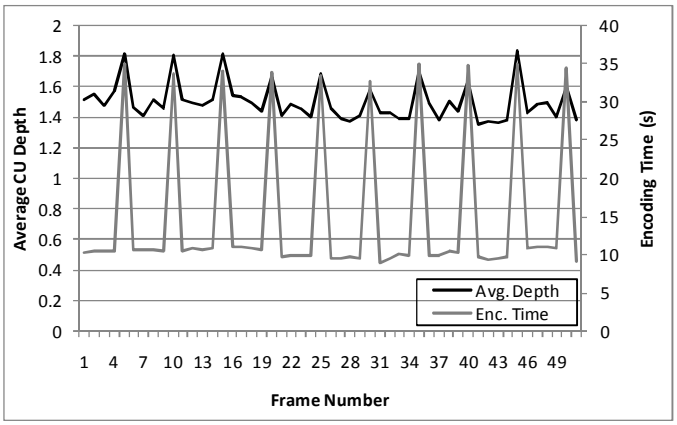

Fig. 10. Average CU depth and encoding time (seconds) per frame for the *BQTerrace* sequence, 60% target complexity.

Tables II-IV present global performance results, including the actual running complexity, bit rate and luminance PSNR for each video sequence at the four target complexities established before. The relative increase/decrease in bit rate and PSNR compared to the case of 100% complexity are also presented. The *Running Complexity* column shows that the algorithm is capable of keeping the running complexity close to the specified target, confirming its effectiveness.

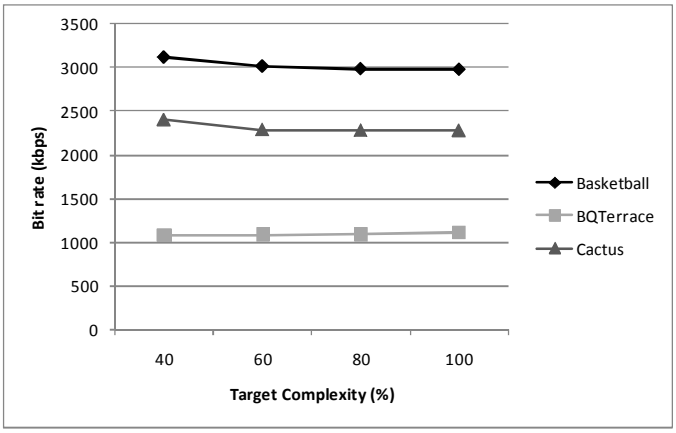

Fig. 11. Bit rate (kbps) obtained for each target complexity.

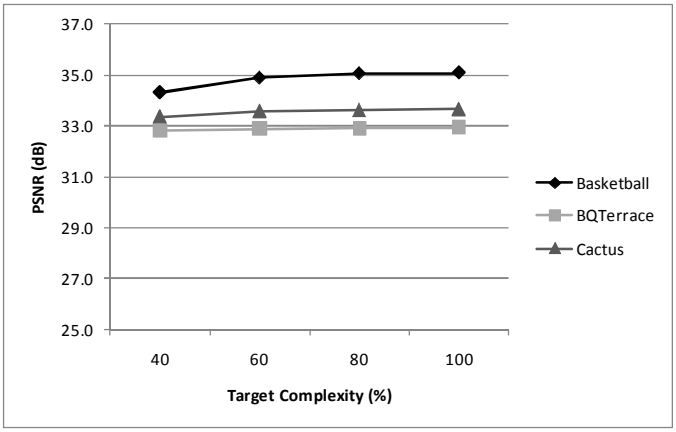

Fig. 12. Luminance PSNR (dB) obtained for each target complexity.

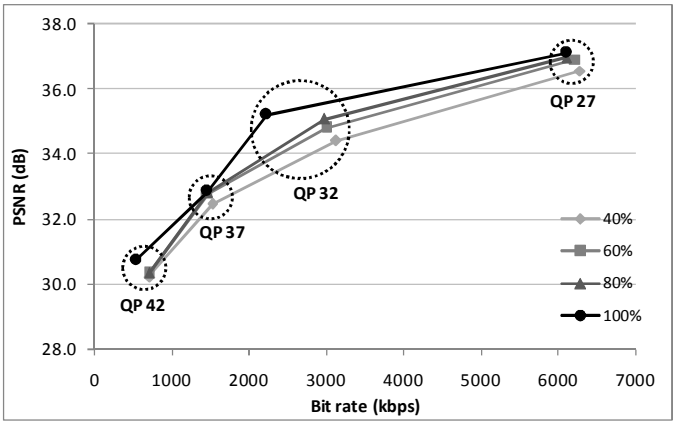

Fig. 13. Luminance PSNR (dB) versus bit rate results for each target complexity (*Basketball* sequence – QPs 27, 32, 37 and 42).

An observation of Tables II-IV reveals a common increase of bit rate for lower target complexities, especially so in the 40% complexity case. This is due to the fact that limiting the number of possible CU sizes for complexity control leads, on the one hand, to a lower number of small CUs than in the case of no complexity control, which results in a larger amount of residual data to be encoded. On the other hand, it is important to notice that CU depth constraining also leads to the use of less CUs. Since the image area of one large CU includes that of several smaller ones, this results in decreasing the amount of bits spent to encode CU header information (coding modes, tree structure, PU structure, TU structure etc).

Overall, these results show that the proposed method is quite accurate in controlling the computational complexity of high efficiency video encoding, exhibiting good RD performance.

TABLE II
RUNNING COMPLEXITY, BIT RATE AND LUMINANCE PSNR FOR EACH TARGET COMPLEXITY (*BASKETBALL*)

Target Complex. (%)	Running Complex. (%)	Bit rate (kbps)	PSNR (dB)	Δ Bit rate (%)	Δ PSNR (dB)
100%	100%	2,978	35.09	–	–
80%	75%	2,985	35.06	+ 0.2	- 0.03
60%	59%	3,015	34.88	+ 1.2	- 0.21
40%	37%	3,124	34.32	+ 4.9	- 0.77

TABLE III
RUNNING COMPLEXITY, BIT RATE AND LUMINANCE PSNR FOR EACH TARGET COMPLEXITY (*BQTERRACE*)

Target Complex. (%)	Running Complex. (%)	Bit rate (kbps)	PSNR (dB)	Δ Bit rate (%)	Δ PSNR (dB)
100%	100%	1,111	32.96	–	–
80%	72%	1,090	32.91	+ 1.8	- 0.05
60%	67%	1,087	32.91	+ 2.1	- 0.05
40%	43%	1,080	32.82	+ 2.8	- 0.14

TABLE IV
RUNNING COMPLEXITY, BIT RATE AND LUMINANCE PSNR FOR EACH TARGET COMPLEXITY (*CACTUS*)

Target Complex. (%)	Running Complex. (%)	Bit rate (kbps)	PSNR (dB)	Δ Bit rate (%)	Δ PSNR (dB)
100%	100%	2,277	33.66	–	–
80%	76%	2,280	33.63	+ 0.1	- 0.03
60%	61%	2,288	33.59	+ 0.5	- 0.07
40%	41%	2,406	33.38	+ 5.7	- 0.28

VII. CONCLUSION

A novel complexity control method for high efficiency video coding was described in this paper. The proposed method relies on the new data structures of the emerging standard HEVC, which resorts to high complexity coding tools and methods to achieve increased efficiency. The experimental results show that for a wide range of complexity reduction (from 80% to 40%) the PSNR drop is lower than 0.8 dB at the expense of a maximum bit rate increase lower than 5.7%.

Overall, the results show that coding complexity can be reduced to a predefined target within acceptable tolerance at the cost of negligible loss of RD efficiency. By setting an upper bound for the computational complexity of high efficiency video encoders, the proposed method is useful in power-constrained portable multimedia devices to reduce energy consumption and to extend the battery life.

REFERENCES

[1] W. Kim, J. You, and J. Jeong, "Complexity control strategy for real-time H.264/AVC encoder," *IEEE Transactions on Consumer Electronics,* vol. 56, pp. 1137-1143, 2010.

[2] C. J. Lian, S. Y. Chien, C. P. Lin, P. C. Tseng, and L. G. Chen, "Power-aware multimedia: concepts and design perspectives,"

IEEE Circuits and Systems Magazine, vol. 7, pp. 26-34, 2007.

[3] L. Xiaoan, W. Yao, and E. Erkip, "Power efficient H.263 video transmission over wireless channels," in *IEEE International Conference on Image Processing*, 2002, pp. I-533-I-536 vol.1.

[4] P. Agrawal, C. Jyh-Cheng, S. Kishore, P. Ramanathan, and K. Sivalingam, "Battery power sensitive video processing in wireless networks," in *9th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1998, vol. 1, pp. 116-120.

[5] A. K. Katsaggelos, Z. Fan, Y. Eisenberg, and R. Berry, "Energy-efficient wireless video coding and delivery," *IEEE Wireless Communications,* vol. 12, pp. 24-30, 2005.

[6] ISO/IEC-JTC1/SC29/WG11, "Joint call for proposals on video compression technology," ed. Kyoto, Japan: Video Coding Experts Group (VCEG), 39th Meeting, 2010.

[7] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 13, pp. 560-576, 2003.

[8] ISO/IEC-JTC1/SC29/WG11, "Vision, applications and requirements for High-Performance Video Coding," ed. Kyoto, Japan: Video Coding Experts Group (VCEG), 39th Meeting, 2010.

[9] ISO/IEC-JTC1/SC29/WG11, "HEVC reference software manual," ed. Geneva, Switzerland, 2011.

[10] ISO/IEC-JCT1/SC29/WG11, "Common test conditions and software reference configurations," ed. Daegu, Korea, 2011.

[11] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine,* vol. 15, pp. 74-90, 1998.

[12] P. Jain, A. Laffely, W. Burleson, R. Tessier, and D. Goeckel, "Dynamically parameterized algorithms and architectures to exploit signal variations," *The Journal of VLSI Signal Processing,* vol. 36, pp. 27-40, 2004.

[13] E. Akyol, D. Mukherjee, and L. Yuxin, "Complexity control for real-time video coding," in *IEEE International Conference on Image Processing*, 2007, pp. I - 77-I - 80.

[14] S. Mietens, P. H. N. de With, and C. Hentschel, "Computational-complexity scalable motion estimation for mobile MPEG encoding," *IEEE Transactions on Consumer Electronics,* vol. 50, pp. 281-291, 2004.

[15] C. Man-Yau and S. Wan-Chi, "Computationally-scalable motion estimation algorithm for H.264/AVC video coding," *IEEE Transactions on Consumer Electronics,* vol. 56, pp. 895-903, 2010.

[16] S. Li, L. Yan, W. Feng, L. Shipeng, and G. Wen, "Complexity-constrained H.264 video encoding," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 19, pp. 477-490, 2009.

[17] Z. Li and G. Wen, "Reusable architecture and complexity-controllable algorithm for the integer/fractional motion estimation of H.264," *IEEE Transactions on Consumer Electronics,* vol. 53, pp. 749-756, 2007.

[18] S. Saponara, M. Casula, F. Rovati, D. Alfonso, and L. Fanucci, "Dynamic control of motion estimation search parameters for low complex H.264 video coding," *IEEE Transactions on Consumer Electronics,* vol. 52, pp. 232-239, 2006.

[19] D. Martinez-Enriquez, A. Jimenez-Moreno, and F. Diaz-de-Maria, "An adaptive algorithm for fast inter mode decision in the H.264/AVC video coding standard," *IEEE Transactions on Consumer Electronics,* vol. 56, pp. 826-834, 2010.

[20] R. Jianfeng, N. Kehtarnavaz, and M. Budagavi, "Computationally efficient mode selection in H.264/AVC video coding," *IEEE Transactions on Consumer Electronics,* vol. 54, pp. 877-886, 2008.

[21] I. R. Ismaeil, A. Docef, F. Kossentini, and R. K. Ward, "A computation-distortion optimized framework for efficient DCT-based video coding," *IEEE Transactions on Multimedia,* vol. 3, pp. 298-310, 2001.

[22] C. Ming-Chen, H. Jhen-Yuan, and C. Pao-Chi, "Complexity control for H.264 video encoding over power-scalable embedded systems," in *13th IEEE International Symposium on Consumer Electronics, 2009. ISCE '09.*, 2009, pp. 221-224.

[23] H. Zhihai and C. Chang Wen, "From rate-distortion analysis to resource-distortion analysis," in *International Conference on Wireless Networks, Communications and Mobile Computing*, 2005, pp. 1527-1532 vol.2.

[24] J. Stottrup-Andersen, S. Forchhammer, and S. M. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in H.264/MPEG-4 AVC," in *IEEE International Conference on Image Processing*, 2004, pp. 111-114 vol. 1.

[25] H. Zhihai, L. Yongfang, C. Lulin, I. Ahmad, and W. Dapeng, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 15, pp. 645-658, 2005.

[26] H. Zhihai, C. Wenye, and C. Xi, "Energy minimization of portable video communication devices based on power-rate-distortion optimization," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 18, pp. 596-608, 2008.

[27] L. Yongfang, A. Ishfaq, and W. Xiaohui, "Adaptive techniques for simultaneous optimization of visual quality and battery power in video encoding sensors," in *IEEE International Conference on Image Processing*, 2006, pp. 2477-2480.

BIOGRAPHIES

Guilherme Corrêa (M'08) received the B.S. degree in Computer Science from the Federal University of Pelotas, Pelotas, RS, Brazil, in 2009 and the M.S. degree in Computer Science from the Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, in 2010. Now he is pursuing the Ph.D. degree in Electrical and Computer Engineering from the University of Coimbra, Portugal. He is a research assistant at the Institute for Telecommunications, Coimbra, Portugal, where he works on video coding algorithms and digital systems design. He is also a student member of the IEEE Circuits and System Society and of the Brazilian Computer Society (SBC).

Pedro A. Assunção (M'98) received the Licenciado and MSc degrees in Electrical Engineering from the University of Coimbra, Portugal, in 1988 and 1993, respectively, and the PhD 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 seventy scientific/technical papers in conferences and journals, three book chapters and three US patents. His current research interests include 2D and 3D video coding, adaptation to diverse networking and user environments, multiple description coding, power-aware video coding, audiovisual error concealment and perceptual quality evaluation.

Luciano Volcan Agostini (M'06-SM'11) received the B.S. degree in Computer Science from the Federal University of Pelotas, RS, Brazil, in 1998 and the M.Sc. and Ph.D. degrees from the Federal University of Rio Grande do Sul, RS, Brazil, in 2002 and 2007, respectively. He is a Professor since 2002 at the Center of Technological Development (CDTEC) of Federal University of Pelotas, where he leads the Group of Architectures and Integrated Circuits (GACI). He has more than 100 published papers in journals and conference proceedings. His research interests include video coding, arithmetic circuits, FPGA based design and microelectronics. Dr. Agostini is a Senior Member of IEEE and he is a member of the IEEE Circuits and System Society, of the IEEE Consumer Electronics Society, of the Brazilian Computer Society (SBC) and of the Brazilian Microelectronics Society (SBMicro).

Luis A. da Silva Cruz (M'11) received the Licenciado and M.Sc. degrees in Electrical Engineering from the University of Coimbra, Portugal, in 1989 and 1993, a MSc 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 of Coimbra where he works on video processing and coding and wireless communications.