Adaptive Fast Quadtree Level Decision Algorithm for H.264 to HEVC Video Transcoding

Antonio Jesús Díaz-Honrubia, *Student Member, IEEE*, José Luis Martínez, Pedro Cuenca, José Antonio Gamez, and José Miguel Puerta

***Abstract*— High Efficiency Video Coding (HEVC) was developed by the Joint Collaborative Team on Video Coding to replace the current H.264/Advanced Video Coding (AVC) standard, which has dominated digital video services in all segments of the domestic and professional markets for over ten years. Therefore, there is a lot of legacy content encoded with H.264/AVC, and an efficient video transcoding from H.264/AVC to HEVC will be needed to enable gradual migration to HEVC. In terms of rate–distortion (RD) performance, HEVC roughly doubles the RD compression performance of H.264/AVC at the expense of a high computational cost. HEVC adopts a quadtree-based coding unit (CU) block partitioning structure that is flexible in adapting various texture characteristics of images. However, this causes a dramatic increase in computational complexity due to the necessity of finding the best CU partitions. This paper presents an adaptive fast quadtree level decision algorithm that is designed to exploit the information gathered at the H.264/AVC decoder in order to make faster decisions on CU splitting in HEVC using a Naïve–Bayes probabilistic classifier that is determined by a supervised data mining process. The experimental results show that the proposed algorithm can achieve a good tradeoff between coding efficiency and complexity compared with the anchor transcoder; moreover, it outperforms other related works available in the literature.**

***Index Terms*— Coding unit (CU) splitting, H.264/Advanced Video Coding (AVC), High Efficiency Video Coding (HEVC), software partitioning, transcoding.**

I. Introduction

IN APRIL 2013, the High Efficiency Video Coding (HEVC) standard [1] was finalized by the Joint Collaborative Team on Video Coding (JCT-VC). More recently, in October 2014, the second edition of the HEVC standard [2] was completed with three important extensions, which include support for more color formats and higher bit depths (Range Extension), support of spatial and fidelity scalability (Scalable High Efficiency Video Coding Extension), and support of multiview video coding (MultiView High Efficiency Video Coding Extension).

The new HEVC standard is a natural evolution of its predecessor, namely, H.264/MPEG4 part 10—Advanced Video Coding (AVC) standard [3]. HEVC was initially conceived with the purpose of not only achieving a highly efficient performance for delivering high-quality multimedia services over bandwidth-constrained networks but also giving support to ultrahigh definition (so-called 4k, 3840 × 2160 pixels, and 8k, 7680 × 4320 pixels), which demand a high bandwidth. While HEVC can bring respite to content producers, aggregators, distributors, and consumers with a higher quality content at the same bitrate, the adoption curve could still be years away.

In terms of rate–distortion (RD) performance, HEVC roughly doubles the RD compression performance of H.264/AVC, but at the cost of extremely high computational and storage complexities during encoding [4]. Among other features, HEVC includes multiple new coding tools, namely, highly flexible quadtree coding block partitioning, which includes new concepts such as coding unit (CU), prediction unit (PU), and transform unit (TU) [5].

Considering both the superior compression performance of HEVC and the large body of content that is currently encoded using the H.264/AVC standard, a transcoder that can convert H.264/AVC bitstreams into HEVC bitstreams is of great value in many applications, especially before dedicated HEVC encoder systems become widely available, while at the same time, various software-based HEVC decoders have been demonstrated [6]. Furthermore, there is a wide availability of H.264/AVC encoders on the market with a good tradeoff in terms of RD performance and low cost. Thus, an H.264/AVC encoder working in tandem with an efficient H.264/AVC to HEVC transcoder may provide a cost-effective means of performing HEVC encoding for many applications in the absence of dedicated HEVC encoders. Therefore, there is a double motivation for an H.264/AVC to HEVC transcoder: on the one hand, to provide interoperability for the legacy video encoded with H.264/AVC when new devices using HEVC emerge and, on the other hand, to take advantage of the superior RD performance of the HEVC standard.

However, all the previously mentioned new coding tools involve a considerable increase in the encoding time in HEVC. With this challenge in mind, this paper presents a soft computing approach, which we have called Adaptive Fast Quadtree Level Decision (AFQLD), which aims to

Manuscript received October 14, 2014; revised March 17, 2015 and June 26, 2015; accepted August 13, 2015. Date of publication August 26, 2015; date of current version January 6, 2015. This work was supported in part by the Ministry of Economy within Competitiveness and European Commission through FEDER Funds under Project TIN2012-38341-C04-04 and Project TIN2013-46638-C3-3-P and in part by the Spanish Ministry of Education, Culture and Sports under Grant FPU 12/00994. This paper was recommended by Associate Editor G. J. Sullivan.

A. J. Díaz-Honrubia, J. L. Martínez, and P. Cuenca are with the High Performance Networks and Architectures Group, University of Castilla–La Mancha, Albacete 13071, Spain (e-mail: antonio.dhonrubia@uclm.es; joseluis.martinez@uclm.es; pedro.cuenca@uclm.es).

J. A. Gamez and J. M. Puerta are with the Intelligent Systems and Data Mining Group, University of Castilla–La Mancha, Albacete 13071, Spain (e-mail: jose.gamez@uclm.es; jose.puerta@uclm.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2473299

exploit the information gathered in the H.264/AVC decoder in order to assist decisions on CU splitting in HEVC using a statistical Naïve–Bayes (NB) classifier to avoid an exhaustive rate–distortion optimization (RDO) search over all possible CU sizes and their modes. Adaptive refers to the fact that the algorithm can dynamically be adapted to the content of each sequence. In an offline data mining process, all the knowledge needed is extracted from the H.264/AVC decoding statistics by means of machine learning (ML) techniques, and is then converted into mathematical models that can be executed in the on-line transcoding process. In other words, our proposal substitutes the brute force scheme used in HM implementation with a low complexity algorithm based on an NB classifier. The experimental results show that the proposed algorithm, compared with the anchor transcoder, can achieve speedups of 2.5× on average over the full set of the HEVC common test sequences, with about a 4% loss in efficiency in terms of Bjontegaard delta rate (BD rate) [7], which measures the increment in bitrate while maintaining the same objective quality.

The remainder of this paper is organized as follows. Section II includes technical background to the new HEVC standard, focusing on coding block partitioning, while Section III identifies the related work being carried out on the topic. Section IV introduces our proposed AFQLD algorithm, and the experimental results are given in Section V. Section VI concludes this paper.

II. Technical Background

HEVC can be considered an evolution of the current H.264/AVC, since it maintains the same block-based hybrid approach used in all previous video compression standards. In addition, new tools have been introduced in HEVC that increase its coding efficiency compared with H.264/AVC, such as the new CU partitioning based on a hierarchical block structure named the coding tree unit (CTU), new transform sizes of 16×16 and 32×32, and a new tool in the decoding loop called sample adaptive offset, which is applied to the reconstructed samples after the deblocking filter with the goal of improving the perceived quality of the decoded sequence. Due to space constraints, a more complete description of these tools and a general HEVC architecture description can be found in [5].

One of the most important changes affects picture partitioning. HEVC defines a new flexible CTU structure [8], which is a replacement of the CU based on nonoverlapping 16×16 pixel blocks (MacroBlocks), as was defined in the previous video coding standards. With the aim of achieving an optimal adaptation of the CU to the content details, the CTU size can vary from a size of 64×64 pixels to a size of 16×16 pixels, and each CTU can iteratively be partitioned into four square sub-blocks of half resolution, named CUs, with a minimum allowable size of 8×8 pixels. Therefore, a CTU can be further partitioned into four depth levels, from $d = 0$ (CU size of 64×64) to $d = 3$ (CU size of 8×8), having 4^d CUs in each depth level. Thus, a CU in depth level d can be denoted by $\mathrm{CU}_{d,k}$ $(k = 0, 1, \ldots, 4^d - 1)$.

HEVC increases CTU flexibility, each $\mathrm{CU}_{d,k}$ becoming a root of two new trees containing two new unit types: PUs, and the TUs. For intra-picture prediction, a PU uses the same $2N \times 2N$ size as for the $\mathrm{CU}_{d,k}$ to which it belongs, allowing it to be split into quad $N \times N$ PUs only for CUs at the minimum depth level. Therefore, the PU size can range from 64×64 to 4×4 pixels. For inter-picture prediction, several nonsquare rectangular block shapes are available in addition to square ones, allowing eight different PU sizes ($2N \times 2N$, $2N \times N$, $N \times 2N$, $N \times N$, $2N \times nU$, $2N \times nD$, $nL \times 2N$, and $nR \times 2N$). The prediction residue obtained in each PU is transformed using the residual quadtree structure [9], which supports various TU sizes from 32×32 to 4×4. In Fig. 1, an example of the partitioning is shown, depicting how a CTU is structured in a hierarchical tree where each CU branch ends in a leaf ($\mathrm{CU}_{d,k}$), which is the root for the two new prediction and transform trees, containing the PU and TU trees.

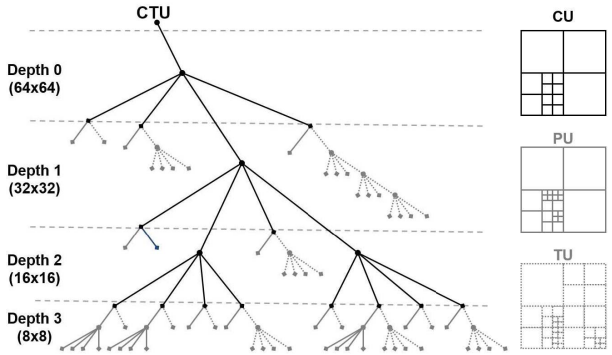


Fig. 1. Partitioning of CTU into CUs, PUs, and TUs.

HEVC checks most of the PUs (inter and intra modes) to decide whether it should split a CU or not by choosing the best RD case by computing the well-known RDO model [10], which requires the evaluation of the total number of possible combinations. This means that the RDO model needs to evaluate the total number of available CUs sizes for the CTU, the total number of prediction modes for each CU, and also the total number of TU sizes. Furthermore, in the case of inter prediction, for each of these PU partitions, a motion estimation (ME) algorithm is called, and in the case of intra prediction, for each of these PU partitions, a total of 35 different coding modes are evaluated. As expected, the selection of the optimal partitioning for each of the three trees is an intensive time-consuming process due to the huge number of combinations that have to be evaluated in order to achieve the best performance [11], [12].

To reflect the practical importance of reducing the HEVC encoder complexity, three optional fast mode decision schemes have been adopted in HEVC reference software, called HEVC test model (HM) [13]. For fast PU decision, the early skip detection (ESD) [14] method and the coded block flag (CBF) fast mode (CFM) [15] method terminate remaining PU decision processes when predefined specific conditions are met. In other words, the ESD checks inter $2N \times 2N$ first and terminates if motion vector (MV) difference is zero and CBF is equal to zero (i.e., values of luma and two chromas are zero after encoding current PU mode). The CFM terminates if PU has CBF equal to zero. On the other hand, the early CU termination (ECU) [16] method terminates the CU splitting if skip is

determined as the best mode of current CU. In this way, a fast decision of CU depth is achieved. The proposed optional fast mode decision schemes reduce the HEVC encoder complexity by reducing the coding efficiency.

As mentioned previously, in the current HEVC reference software, in order to encode a given CU, the encoder tries all possible prediction modes. It is obvious that this scheme, by trying all and selecting the best, obtains the best partitioning of CU in RD terms at the expense of an extremely high computational complexity.

The complexity analysis described in [11] and [12] shows that examining all possible modes is the most time-consuming part of the total encoding process. Based on these observations, we present an innovative approach to jointly optimize the decision mode and ME. In our proposal, the CTU splitting computation problem is posed as a data classification problem where the information gathered in the H.264/AVC decoder assists decisions on CU splitting using a statistical NB classifier to avoid an exhaustive RDO search over all possible CU sizes and its possible modes, as mentioned in the previous sections. Unlike the reference software where MVs are estimated for all inter-mode PU types, in our approach, no ME is required for a particular CU size if that size is not selected by the statistical NB classifier. In the same way, no intra-prediction is required for a particular CU size if that size is not selected by the statistical NB classifier. The proposed approach will perform the ME and the intra prediction only for all the possible PU types belonging to the final CU size determined by the classifier, considerably reducing the huge RDO complexity in the CTU splitting algorithm of the HEVC.

III. Related Work

Video transcoding is the process of converting a compressed video stream encoded with a given format or characteristics into another video stream encoded with a different codec or features. The simplest transcoding process should perform the complete process of decoding and fully re-encoding [17], but this is not time effective. Proposals available in the literature try to avoid unnecessary operations in the second part of the transcoder (encoding stage) or even to accelerate complex modules simply using information collected in the decoding process as part of the transcoder.

In the framework of this paper, there are currently few approaches that deal with the problem of converting streams already encoded in H.264/AVC into the new HEVC standard. The first approach was proposed in 2012 and presented in [18], focusing on reducing the number of CU and PU partitions to be checked by means of an improved RDO metric. The time reduction achieved is an acceptable 70%–80% at the expense of a great RD penalty of 30%. In the same year, Peixoto and Izquierdo [19] proposed the reuse of MVs as well as a similarity metric to decide which HEVC CU partitions should be tested. This proposal obtains a maximum of 4.13× speedup with a rate penalty up to 10.92%. One year later, Peixoto *et al.* [20] proposed two alternatives to map H.264/AVC MBs into HEVC CUs based on an ML model: one of them works offline and the other uses a dynamic training on-line stage. An extension of [19] and [20] was published in [21], where the first k frames of the sequence are used to compute the parameters so that the transcoder can learn the mapping for that particular sequence. Then, two different types of mode mapping algorithms are proposed. In the first solution, a single H.264/AVC coding parameter is used to determine the outgoing HEVC partitions using dynamic thresholding. The second solution uses linear discriminant functions to map the incoming H.264/AVC coding parameters to the outgoing HEVC partitions; this solution is called proposed transcoder for content modeling using linear discriminant functions (PTCM-LDF) and will be used in this paper to compare our approach with related works. The first solution obtains a tradeoff between the speedup and bitrate increases of 3.08× and 16.2%, respectively. More recently, Peixoto *et al.* [22] have proposed a further step in the framework of H.264/AVC to HEVC transcoding; the proposal deals with a solution also based on ML to map H.264/AVC MBs into HEVC CUs and a statistical model of the HEVC RD model to perform an early termination, obtaining on average a speedup to 3.83× with a rate and distortion penalty of around 4%.

Other approaches available in the literature deal with H.264/AVC to HEVC transcoding for surveillance by considering the long static background characteristics of surveillance videos [23]. Jiang *et al.* [24] proposed a transcoder algorithm based on region feature analysis. The main idea consists in dividing each frame into three regions in terms of CTU on the basis of the correlation between image complexity and the coding bits of the H.264/AVC source bitstream. The results obtained in terms of speedup and BD rate are, on average, 1.93× and 1.73%, respectively.

Finally, there are other related transcoding-based approaches available in the literature that involve the HEVC standard, but they do not focus on H.264/AVC to HEVC. For example, Van *et al.* [25] and De Praeter *et al.* [26] proposed a framework for bitrate adaptation depending on the behavior of the network. Another scheme available in the literature in the framework of HEVC-based transcoding is presented in [26], where a spatially misaligned HEVC transcoding is proposed. Another HEVC-based transcoder scheme is presented in [27], where an MPEG-2 to HEVC transcoder is proposed based on content modeling.

The authors of this paper have already published some work related to this field. In fact, in [28], a multiple frame transcoding algorithm is proposed in which frames used in H.264/AVC as reference for P or B predictions are the only frames checked in the HEVC ME algorithm. And more recently, in [29], a preliminary version of this paper was presented, which was called fast quadtree level decision (FQLD). In this work, an algorithm to determine CU depth based on a Bayesian probabilistic model is proposed, although some improvements and refinements have now been implemented with respect to this paper.

1) A new online threshold procedure to determine whether a CU should be split.
2) The HEVC information is also used in the Bayesian model and in that incoming from H.264/AVC.
3) New models based on each temporal layer and frame type (P or B).

4) A new classification algorithm for a CU size of 16×16 pixels.
5) The proposal has been migrated to the newest version of HM reference software available at this moment.
6) A full performance evaluation has been carried out.

A comparison between the proposed and the previous approach can be seen in Section V-D. Moreover, the authors of this paper would like to emphasize the improvements achieved using a Bayesian model instead of decision trees as they is the most commonly used option in [19]–[21]. These are basically the following.

1) Models based on decision trees are more complex on the training stage (exponential against linear with respect to data, in the worst case).
2) Once new training data are generated, updating the model is not so easy as in the NB model; NB models are linear on the number of data for training and also the number of variables in classification time.
3) Models based on decision rules are not suitable for adapting dynamic thresholds that can be useful to adapt model/probabilities based on video content.

IV. Data-Driven Probabilistic CTU Splitting Algorithm

As mentioned above, the HEVC encoding process can take an immensely long time. But as can be seen in the previous section, there are also some techniques that can reduce the time needed for transcoding to previous standards and, even, to HEVC. Most of these techniques are based on the use of data mining techniques [30]. In particular, mainly decision trees and decision rules have been considered. In this paper, we propose the use of probabilistic models, which are capable of intrinsically dealing with the uncertainty inherent to the targeted problem, so providing more flexible classifiers in order to suit the particularities of each video scene.

As a general description, we can say that our objective is to design models that help the transcoder to make the decision of splitting the CU under study and then descend a level in the quadtree or, on the contrary, to make the decision of stopping and choosing the current level as the maximum allowed depth. Our proposal consists of designing this decision function by following a knowledge discovery from data (KDD) process. Thus, our ultimate goal is to learn a (set of) model(s)

$$\mathcal{M}(f_1, f_2, \ldots, f_n) \longrightarrow \{C_S, C_N\}$$

to be plugged into the transcoder. Therefore, our objective corresponds to the well-known supervised classification task, where we must decide on a set of categories, to split the CU (C_S), and descend a level in the quadtree, or not (C_N), and choose the current level as the maximum allowed depth, using the values of a set of features that describe our current problem instance.

Before entering into the details of the KDD process followed, let us give a general description of the proposed CTU splitting algorithm.

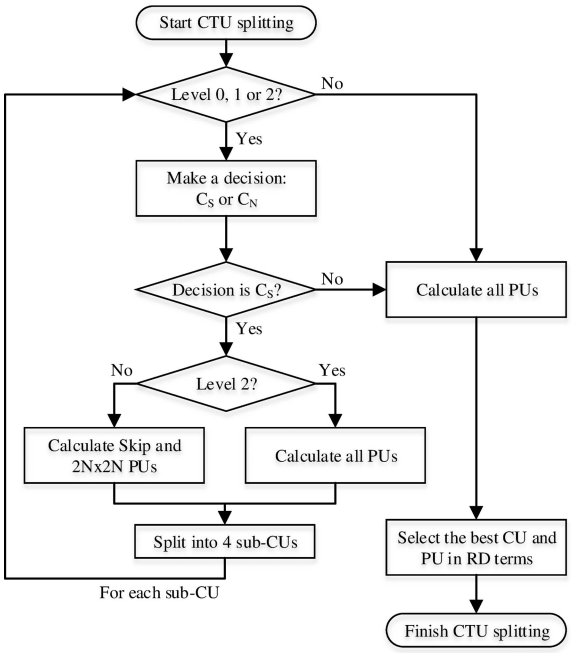


Fig. 2. Diagram of the proposed AFQLD algorithm.

A. General Description of Algorithm

As described above, C_S and C_N are the two categories (or class labels) to be predicted by our decision function or classifier. Furthermore, as the selection can be carried out in a different way depending on the quadtree depth and the type of frame being processed, several classifiers are needed.

If the chosen decision is C_S, we need to take into account some considerations. Thus, if the algorithm obtained a 100% hit rate in all cases, it would be logical to obviate all the PU computations in the levels predicted as C_S. However, this hit rate cannot be obtained in real applications, since each label is associated with a probability of being chosen, so it might occasionally misclassify partitions. With this fact in mind, if C_S is chosen, only skip and $2N \times 2N$ PUs are checked for levels 0 and 1, while all PUs are checked at level 2. On the other hand, if the decision is C_N, then the current depth is considered as final and all PUs at this CU depth are evaluated and the algorithm for this CTU finishes. Thus, if some of the RD costs that were calculated for higher levels are better than the best RD cost for the final level, the quadtree is allowed to return to the best one among those calculated.

Fig. 2 schematically describes the proposed CU splitting algorithm, which we have called AFQLD, where the term adaptive refers to the fact that the algorithm can be dynamically adapted to the content of each sequence, as will be described below.

From the description of the global behavior of the proposed algorithm, it is clear that we need to design a model $\mathcal{M}_l$ for making the decision at each level $l = 0, 1, 2$. In this study, we rely on a KDD-based approach (see Section IV-B) for levels 0 and 1 of the quadtree (CU sizes of 64×64 and 32×32 pixels, respectively), while at level 2, a much simpler strategy is followed. Basically, as CU size at level 2 is 16×16 pixels, we take advantage of the fact that this

Algorithm 1 Overall Transcoding Process

1: Decode H.264/AVC sequence and save each feature F_i
2: Cost[1] = Cost[2] = Cost[3] = Cost[4] = 0
3: **for all** GOPs in sequence **do**
4: **for all** frames in GOP **do**
5: **for all** CTUs in frame **do**
6: **if** First GOP **and** cost[currentFrameEnergy]==0 **then**
7: cost[currentFrameEnergy] += SplittingError
8: **else**
9: **for all** Not finished CUs in CTU **do**
10: **if** Level $\in \{0, 1, 2\}$ **then**
11: **if** Level == 2 **then**
12: **if** MB mode is Skip or 16×16 **then**
13: Classify as C_N
14: **else if** MB mode is 16×8 or 8×16 **then**
15: **if** Adjacent MB modes are Skip, 16×16, 16×8 or 8×16 **then**
16: Classify as C_N
17: **else**
18: Classify as C_S
19: **end if**
20: **else**
21: Classify as C_S
22: **end if**
23: **else**
24: Fetch the needed features $F_1, F_2, \ldots, F_n$
25: Classify as C_S or C_N according to the decision of the appropriate model $\mathcal{M}_l$ using the fecthed features
26: **end if**
27: **if** Decision == C_S **then**
28: **if** Level == 2 **then**
29: Calculate all PUs
30: **else**
31: Calculate Skip and 2Nx2N PUs
32: **end if**
33: Split into 4 sub-CUs
34: **else**
35: Calculate all PUs
36: Select the best CU and PU in RD terms and finish this CU
37: **end if**
38: **else**
39: Calculate all PUs
40: Select the best CU and PU in RD terms and finish this CU
41: **end if**
42: **end for**
43: **end if**
44: **end for**
45: **end for**
46: **end for**

is the MB size in H.264/AVC too, so the proposed algorithm mimics H.264/AVC as it can be seen in lines 12–22 of Algorithm 1.

TABLE I
THEORETICAL EFFICIENCY OF THE AFQLD ALGORITHM

	CU size probability				Avg. RDO operations	RDO op. saving (%)
	64x64	32x32	16x16	8x8		
2560x1600	0.087	0.147	0.402	0.364	1258.58	46.2
1920x1080	0.069	0.137	0.378	0.416	1354.78	42.1
832x480	0.031	0.104	0.342	0.522	1561.73	33.2
416x240	0.019	0.084	0.323	0.574	1660.44	29.0
1280x720	0.221	0.172	0.397	0.211	904.27	61.3

Finally, in order to give an approximation of the expected efficiency for the proposed algorithm, Table I shows the probability of selecting each CU size for the resolutions given in [31] (obtained from sequences that have been fully decoded from H.264/AVC and encoded with HEVC), the number of RDO operations that the proposed algorithm would carry out on average (according to the number of RDO operations carried out at each level), and the percentage of RDO operations that would theoretically be saved.

B. Learning Splitting Model

KDD [32] can be defined as the overall process of finding and interpreting patterns or models in data. The KDD process consists of iterating some of the following steps: creating a target dataset, where we mainly identify the variables related to the targeted problem and select a subset of samples; data cleaning; preprocessing; and data selection, where we devise a subset of variables; and possibly cleaning and transforming them; learning form data, where we have to select the data mining (ML) algorithms and apply them over the selected dataset in order to obtain the models/patterns; model/pattern evaluation/validation; and finally knowledge exploitation.

The problem domain, which is the transcoding problem, has been reviewed in the previous sections. As a rough summary, we can state the following.

1) The task under study can be modeled as a supervised classification problem, where our aim is to predict the correct value for a binary class variable: to split the CU under study (C_S) and not (C_N).
2) Several models must be learnt, depending on the CU depth (0 or 1) and the average energy of the residue, where four levels of energy have been considered (1–4), where 1 represents high residual energy and 4 represents low residual energy. Thus, each frame in RA configuration can be identified with different energy according to its hierarchical layer and frames in low-delay B (LB) and low-delay P (LP) configurations can be associated with levels 3 and 2 of energy, respectively, since LB frames have low energy, but not as low as those with the lowest energy in RA configuration, and LP frames have higher energy than the previous ones.
3) The following families of features can be good predictors to help in decision making.
   a) Features that correctly model the spatial and temporal complexity.
   b) As the framework of this work is a transcoder, information fetched from the decoding stage is available.

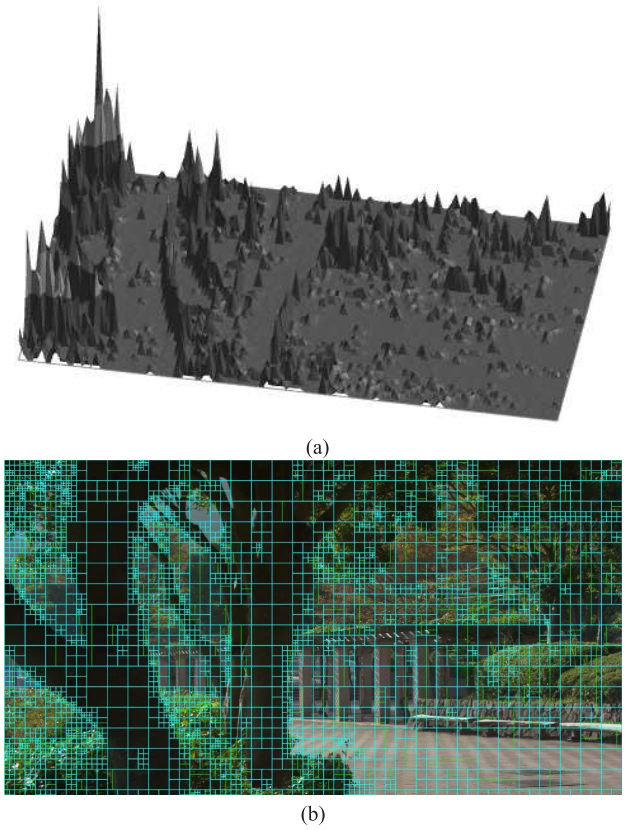


Fig. 3. Visual relationship between CTU splitting in HEVC and the number of bits used to encode the frame in H.264/AVC (QP = 27). (a) Bits per MB in H.264/AVC stream. (b) Original CTU splitting in HEVC.

c) Statistical data, such as the variance of the residue [33], have been shown to work well in previous transcoders.
d) Information that could summarize both the spatial and the temporal information simultaneously.
e) Information from the HEVC coding stage can also be extracted dynamically.

Regarding the variables, we have identified a number of groups of candidate features to be included. This decision comes from our prior knowledge of the problem and also from ad hoc analyses. As an example, the following analysis has been carried out to include the variable w_{Bits} as a representative feature to summarize both spatial and temporal information. The number of bits used to code an MB in the original H.264/AVC stream (w_{Bits}) could also give some information about the Lagrangian cost of each MB in the original sequence. The selection of this feature is based on the study of Fig. 3. In particular, in Fig. 3(a), there is a 3D graphic of the number of bits used to encode each MB from frame 2 in the sequence *ParkScene*, where taller parts of the graphic represent a higher number of bits, while in Fig. 3(b), the CU partitioning after transcoding is shown. It can be seen at a glance that taller parts of the graphic (that is, higher numbers of bits used to encode an MB) correspond to parts of the frame where the CUs are smaller, so there seems to be some kind of correspondence between the number of bits used to encode an MB in H.264/AVC and the CTU partitioning in HEVC. One can even make out the shape of the trees in the frame as hills between plains in the 3D graphic.

This type of analysis has been carried out with different features in order to decide whether to include them or not in our initial dataset. In case of doubt, the mere suspicion that a variable may be correlated with the CTU splitting is enough for us to include it, since it will be discarded later during the data selection process if considered redundant or irrelevant for the task considered.

After this study, each feature was fetched from the H.264/AVC stream or from the HEVC encoding process. In the case of data extracted from the H.264/AVC stream, to determine the correspondence between the different partitions of both standards, for each CU, an overlap of the MBs in the CU covering area was made, e.g., a 64 × 64 pixel CU corresponds to 16 MBs and a 32 × 32 pixel CU corresponds to 4 MBs. Thus, our initial set of features, **F**, contains the following 26 variables:

1) w_{QP}: QP value used to encode the stream;
2) w_{Bits}: number of bits used to encode all the MBs for the current CU after applying the context-adaptive binary arithmetic coding operation;
3) w_{Intra}, w_{Skip}, w_{16}, w_4, and w_{Inter}: number of intra, skip, inter 16 × 16, inter 4 × 4 and other inter size MBs, respectively;
4) w_{DCTno0}: number of nonzero DCT coefficients;
5) w_{Width} and w_{Height}: frame width and height, respectively;
6) w_{MVsum}: sum of all the MV components contained in the frame;
7) w_{ResAvg} and w_{ResVar}: average and variance of the residue for the covered area, respectively;
8) $w_{\text{ResAvgSubCU1}}$, $w_{\text{ResAvgSubCU2}}$, $w_{\text{ResAvgSubCU3}}$, and $w_{\text{ResAvgSubCU4}}$: average of the residue for each sub-CU: 1, 2, 3 and 4, respectively;
9) $w_{\text{ResVarSubCUs}}$: variance of the above four values;
10) w_{SobelH} and w_{SobelV}: sum of applying the Sobel operator [34] to the residue in horizontal and vertical directions;
11) w_{MVxAvg}, w_{MVyAvg}, w_{MVxVar}, and w_{MVyVar}: average and variance of x and y MV components, respectively, for the covered area;
12) w_{SkipCost} and $w_{2N\times 2N\text{Cost}}$: Lagrangian cost of choosing skip and $2N \times 2N$, respectively, at the HEVC coding stage.

Regarding the samples, in order to create an adequate data set to train the models, five sequences have been selected from those described by the JCT-VC in [31]. The criterion to choose them has been the use of the spatial index (SI) and temporal index (TI) according to the ITU-T P.910 recommendation [35], so that the chosen sequences cover a wide range of the peculiarities of an image. It is useful to compare the relative SI and TI of the test sequences, since the compression difficulty is usually directly related to both spatial and temporal complexity.

In particular, the *PeopleOnStreet, ParkScene, PartyScene*, and *BQSquare* sequences have been selected. These represent one sequence per class (A, B, C, and D), so that they

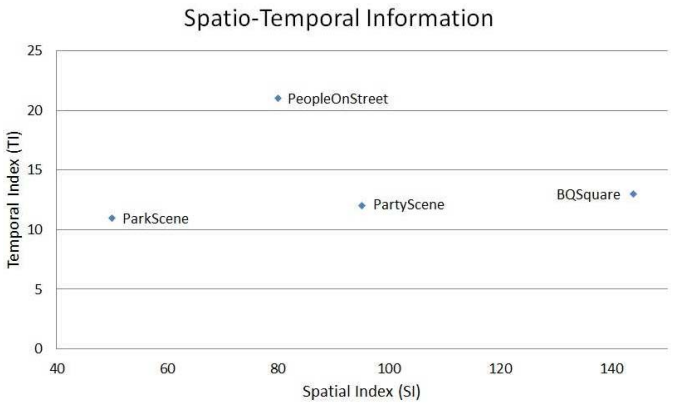


Fig. 4. SI and TI of the selected sequences.

can also be representative of the wide range of resolutions (see Section V-A for more details). Fig. 4 shows the SI and the TI of each sequence. As can be observed, they cover a wide range of peculiarities. The first 1000 CUs of each combination of sequence, QP, CU depth, and temporal layer (with RA configuration and QP values $\{22, 27, 32, 37\}$) after skipping the first frame (since it is coded as intra), were selected as input for the supervised learning process, which results in eight classifiers and a total of 16 000 to learn each of them.

Therefore, the initial dataset can be viewed as a $16\,000 \times 27$ bidimensional matrix, which corresponds to 16 000 instances in which a decision about splitting a CU or not is needed. For each one of these cases, we know the correct decision (C_N or C_S) and the value of the 26 predictive features. Now, our goal is to learn classifiers that are able to make the decision on unseen new cases.

Specifically, a different dataset has been built for each CU depth (0 and 1) and each level of residual energy (1, 2, 3, and 4): $\mathrm{DB}_{i,j}, i \in \{0, 1\}, j \in \{1, 2, 3, 4\}$. For the training, RA configuration has been used, since it contains frames from all levels of residual energy (according to its hierarchical GOP structure). Therefore, this process involves eight different datasets with a total of 128 000 instances.

In the field of the data mining and data analysis, there are a large number of tools devoted to deal and analyze data sets. One of the most popular is Weka [30], which is a tool written in java and it implements the most popular data mining algorithms. Moreover, Weka is open-source software issued under the GNU General Public License. Because of this, we have selected this tool to conduct the data mining tasks in our problem.

C. Preprocessing Data: Discretization

Most of the previous selected features are of numerical nature. Although the selected probabilistic classifier (NB) and the ML algorithm to train it are able to cope with numerical variables, they impose an assumption that the values for a feature projected for each class value follow a univariate Gaussian distribution, which is quite improbable in practice. Because of this, we decided to preprocess the feature by discretizing them.

From the existing range of available discretization techniques, we have selected the entropy-based supervised discretization algorithm proposed by Fayyad and Irani [36]. This algorithm fits well with our task because its supervised nature allows splitting the numerical variable into a set of intervals that yields more discriminative power regarding the class variable. Two well-known advantages of this algorithm are: 1) resulting intervals are of different widths and 2) the number of intervals for each feature is automatically selected by the algorithm (see [36] for details). Once we have the data sets collected in the previous step, they are used as input for the Weka package. In order to apply the Fayyad and Irani discretization algorithm, the supervised.Discretize filter has to be selected in Weka (default parameter setting leads to canonical Fayyad and Irani algorithm [36]).

The output of the discretization process is a new dataset in which all the variables are discrete/categorical, heterogeneously defined regarding the number of intervals and their widths. It should be noted that some of the features ($w_{\mathrm{QP}}, w_{\mathrm{bits}}, w_{\mathrm{width}}, \ldots$) have numerical nature, but they already have discrete values (they belong to natural numbers); in this case, this process is still valid and it splits $\mathbb{N}$ into intervals. An example of this process is shown in Section IV-G.

Using this discretization strategy, all the learnt models have increased their performance. As an example, the accuracy of the NB model learnt increases about 10% for $\mathrm{DB}_{0,2}$ with respect to the use of numerical variables.

D. Feature Subset Selection

Probabilistic classifiers in general, and NB in particular, are quite sensitive to the feature set used to induce the classifier. Thus, the presence in the training set of irrelevant (uncorrelated with the class) variables and/or redundant [correlated with other feature(s)] may significantly affect the precision of the learned classifier. Furthermore, different tasks may require different subsets of features, and this is our case as we need to learn different classifiers (levels and frames).

To cope with the aforementioned problems a feature subset selection (FSS) is applied in order to select the proper subset of features for each particular task [37]. We selected a greedy strategy in combination with wrapper evaluation, as it is computationally efficient and robust against overfitting. The forward selection process starts with the empty set and iteratively incorporates the best remaining feature at each step. In the wrapper approach, the best feature is decided to be the one that, when joined to the current subset of selected features, allows the ML algorithm to induce the classifier with the maximum accuracy among those trained in that iteration. The FSS process finishes when the addition of a new feature no longer improves the accuracy of the induced classifier. The advantage of using a wrapper technique is that the same ML algorithm to be used for model discovering can be used at this step, thus maximizing the knowledge transferred between these two KDD steps.

To cope with this step, we have used again the Weka data mining suite. This time we have used the dataset discretized in the above step as the input data. To do FSS described in the last paragraph in Weka, the filter supervised.AtributeSelection is used. As an attribute subset evaluator, we have used

WrapperSubsetEval coupled with the NB as subrogate classifier. As a search method, we have selected GreedySteepwise instantiated to a sequential forward search.

Following the example for the database $DB_{0,2}$, once the variable selection is carried out, the model accuracy achieves 92%. Note that the initial accuracy, with continuous variables, was about 70%. In all the learnt models, the cardinality of the selected subset has been in the range 2–5, with an average of 3.6. That means an average reduction of 86%.

E. Mining the Data

After the previous steps, we have a higher quality dataset than the one initially selected, and from better data, we will be capable of *mining* better models. Let us first briefly introduce the ML algorithm selected in this paper and then we will outline the data mining process carried out.

1) Naïve–Bayes Classifier: The NB classifier [38], [39] is perhaps the simplest representative of probabilistic classifiers. Despite its simplicity, it has been successfully used in many domains and continues to be one of the most commonly used algorithms in the field of machine learning and data mining. This simplicity is the main reason for its choice in the transcoding algorithm.

NB relies on a strong independence assumption: all features are conditionally independent given the class. This can be a considerable drawback in theory, but in practice, NB shows a competitive performance with respect to more complex ML algorithms. Furthermore, this independence assumption brings the following important advantages: 1) NB is memory and space efficient at both learning and classification; 2) NB has the ability of learning good models even from scarce data; and 3) NB is robust against overfitting, thanks to the small number (and low-order) of parameters (conditional probability tables), it needs to learn.

As a probabilistic classifier, NB inherently deals with the uncertainty present in the data, behaving in this way as a flexible classifier. The classification rule is based on the maximum *a posteriori* probability principle, that is, given the values of the input features for an object $x = (f_1, \ldots, f_n)$, it computes the posterior probability distribution for the class variable C and chooses the class label $\{c_1, \ldots, c_K\}$ with the highest probability

$$c^* = \underset{C \in \{c_1, \ldots, c_K\}}{\operatorname{argmax}} P(C | f_1, \ldots, f_n). \tag{1}$$

Applying Bayes' Theorem and realizing that the denominator is the same for all class labels, we get

$$\begin{aligned} c^* &= \underset{C \in \{c_1, \ldots, c_K\}}{\operatorname{argmax}} \frac{P(f_1, \ldots, f_n | C) P(C)}{P(f_1, \ldots, f_n)} \\ &= \underset{C \in \{c_1, \ldots, c_K\}}{\operatorname{argmax}} P(f_1, \ldots, f_n | C) P(C). \end{aligned}$$

Obviously, managing the joint probability distribution of all the variables involved $P(f_1, \ldots, f_n, C)$ becomes unfeasible even for small datasets (few variables), not only because of the number of parameters needed (exponential in the number of variables) but also because of the amount of data required for the distribution to be representative. However, thanks to the conditional independence assumption considered in the NB model, the joint distribution is approximated by the following factorization:

$$P(f_1, \ldots, f_n, C) = P(C) \prod_{i=1}^{n} P(f_i | C)$$

and so the NB classifier can be formulated as

$$c^* = \underset{C \in \{c_1, \ldots, c_K\}}{\operatorname{argmax}} P(C) \prod_{i=1}^{n} P(f_i | C).$$

Thus, the algorithm to induce the NB model only needs to calculate the marginal distribution for C

$$P(C = c_k) = \#(C = c_k)/N \tag{2}$$

where $\#(C = c_k)$ is the frequency that C is equal to the value c_k in data and N is the total amount of data, and the probability distribution of each feature conditioned to the class $P(f_i|C)$. If f_i is a discrete variable, as it is our case, its conditional probability is computed by counting relative frequencies. This set of frequencies is computed in only one reading of the data, actually the computational complexity in learning and NB is one of the fastest algorithms, which takes $\mathcal{O}(Nn)$, being N the number of instances and n the number of features in database.[1] In addition, NB is linear in classification phase and it takes $\mathcal{O}(n)$.

2) Learning the Models: Starting from the original set of features and the eight datasets, one for each combination of CU depth level (0 or 1) and residual energy level (1, 2, 3, or 4), eight independent data mining processes are carried out to learn the corresponding models ($\mathcal{M}_{ij}, i \in \{0, 1\}$, $j \in \{1, 2, 3, 4\}$). Each model corresponds to the NB classifier induced by taking as training set the dataset obtained by applying discretization followed by FSS for the corresponding initial dataset. Once an NB classifier has been learned using the Weka package, the corresponding model has been implemented, which means that the Weka package is not used during the transcoding process.

F. Model/Pattern Evaluation

When evaluating the learnt models using standard scores such as cross-validated *accuracy* or the area under the receiving operator characteristics (ROC),[2] curve apart from discovering that discretization and FSS improves the models both in terms of simplicity and accuracy, we also realized that our models should be calibrated taking into account the specific problem we are tacking. By calibration, we mean the problem of setting the specific classifier rule. From (1) and taking into account that our class only has two labels, it is easy to see that the induced classification rule is to choose C_N if $P(C_N) > P(C_S)$. Semantically, this fact means that both types of mistake, namely, classifying as C_N when the actual value is C_S or classifying as C_S when the actual value is C_N, have

[1] For example, the models used in related works, such as linear discriminant functions and decision trees, take $\mathcal{O}(Nn^2)$ and $\mathcal{O}(nN \log N) + \mathcal{O}(N(\log N)^2)$, respectively [40]. Moreover, probabilistic algorithms are easily adaptable to each specific instance, as described in Section IV-F.

[2] ROC.

the same *cost*. However, we have detected that this is not the case in the transcoding problem. First, intuitively and generally speaking, the cost of making an error of not splitting when HEVC decides to split should be more costly because if we decide to split, speed is decreased but the quality of the image is preserved. And second, the costs should be dependent not only on the model but also on the video sequence. Therefore, in order to achieve a good performance for the proposed algorithm, the costs of making every decision are dynamically adjusted.

There are four different costs in the classifying process: the cost of correctly choosing C_N (Cost_{NN}), the cost of correctly choosing C_S (Cost_{SS}), the cost of choosing C_N when the correct decision would have been C_S (Cost_{NS}), and the cost of choosing C_S when the correct decision would have been C_N (Cost_{SN}). Once we have estimated these costs, the decision rule should be $P(C_S) \times \text{Cost}_{\text{SN}} > P(C_N) \times \text{Cost}_{\text{NS}}$.

In order to measure this costs given the correct decision, the Lagrangian cost of splitting (L_S) and of not splitting (L_N) have been used, as well as the mathematical concept of absolute error, as shown in (3), where Cost_{ij} represents the cost of choosing C_i when the correct decision is C_j, and L_j is the Lagrangian cost of the correct decision, L_i is the Lagrangian cost of the prediction made by the classifier, where i, $j \in \{S, N\}$

$$\text{Cost}_{ij} = |L_j - L_i| \times \omega_{ij} \tag{3}$$

where ωij is a weight associated for each particular cost. The weight associated with the Cost_{NS} is 2.0 and 1.0 for the rest.[3] It should be noted that $\text{Cost}_{\text{NN}} = \text{Cost}_{\text{SS}} = 0$ as expected, since the cost of not making a mistake must be 0. On the other hand, all the costs can be normalized by dividing them by Cost_{NS} (Cost_{SN} could also have been chosen), so the normalized costs are the following: $\overline{\text{Cost}_{\text{NS}}} = 1$, $\overline{\text{Cost}_{\text{SN}}} = \text{Cost}_{\text{SN}}/\text{Cost}_{\text{NS}}$. Taking this fact into account, the rule will depend only on one value: Cost_{SN} normalized by Cost_{NS}, since all the others are constants. Therefore, the decision rule is transformed into $P(C_S) \times \overline{\text{Cost}_{\text{SN}}} > P(C_N)$.

As mentioned above, these costs are estimated for each sequence and each particular model (they are only learned at the beginning of the sequence; however, they could be learned again every n frames or when a scene change is detected, but are out of the scope of this paper). For the calculation process, the first frame of the current residual energy level is encoded by running the algorithm but letting the HEVC encoder make the correct decision so that it can calculate the actual and the predicted decision.[4] If Cost_{SN} or Cost_{NS} are 0, then the following frame is used to calculate the threshold, and so on (setting the process not to exceed the first GOP). In this way, the time spent to encode a few frames (with the maximum of a GOP) without the AFQLD algorithm is negligible compared with the whole encoding time, while letting us have a dynamic sequence content and QP-dependent cost and threshold.

[3]Note that this choice was totally heuristic. We have empirically tested several weights for $C_{\text{NS}} \in \{1, 2, 3, 4, 5\}$, and the best results were obtained with 2.0. Also note that this fact is in agreement with our intuition so as to preserve the quality of the image.

[4]Therefore, this adaptive scheme is carried out online in transcodification time by using the models implemented in the transcodification software.

TABLE II
MODEL OF w_{Bits} FEATURE IN $M_{1,4}$ CLASSIFIER

	$\#(w_{Bits} \in \cdot \vert C_S)$	$\#(w_{Bits} \in \cdot \vert C_N)$
$[0, 0]$	2214	2029
$(0, 46]$	3018	4382
$(46, 108]$	1051	976
$(108, 404]$	1207	617
$(404, +\infty)$	423	93
Total	7913	8097

G. Model/Pattern Interpretation and Knowledge Exploitation

The last stage of the KDD cycle is devoted to analysis/interpretation of the models learned and their exploitation in the final system. Here, we briefly describe one of the four discovered models, while the performance evaluation of the system integrating the discovered models deserves, in our opinion, a whole section (Section V).

We will now comment on some parts of the classifier learnt for residual energy level 4 and level 1 of the quadtree ($\mathcal{M}_{1,4}$). The final set of features, after discretizing and the selection of features subset, is the following:

1) w_{Bits}: $[0, 0]$, $(0, 46]$, $(46, 108]$, $(108, 404]$, $(404, +\infty)$;
2) w_{MVsum}: $[0, 14131]$, $(14131, 15064]$, $(15064, 16529]$, $(16529, 64264]$, $(64264, 2110792]$, $(2110792, +\infty)$;
3) w_{MVyVar}: $[0, 6.248047]$, $(6.248047, +\infty)$;
4) w_{SkipCost}: $[0, 3337]$, $(3337, 25743]$, $(25743, 274594]$, $(274594, +\infty)$.

It can be seen, as expected, that the w_{Bits} feature was selected. To give an example of the information which the classifier must store, Table II shows the number of instances for each interval of the w_{Bits} feature. The probability can be obtained by dividing each entry in Table II by the sum of all the elements in its column, i.e., $P(w_{\text{Bits}} \in (0, 46] | C_S) = 3018/7913 \approx 0.38$. The tables for the remaining variables are similar to Table II, but they are not included here due to space limitations.

H. Overall Transcoding Process

The overall transcoding process is shown in Algorithm 1. First, the H.264/AVC sequence is decoded (and all the needed information is generated) and the costs described in Section IV-F are set up to 0 for each level of energy (from 1 to 4). Then the HEVC encoding process is started, encoding each GOP using the AFQLD algorithm (Fig. 2) to encode the CTUs in the frames. It can be noted that the first GOP is used to learn the costs.

V. PERFORMANCE EVALUATION

This section aims to evaluate the AFQLD algorithm presented in this paper by showing its performance with different configurations and applying it to different depths of the quadtree. A comparison with another relevant state-of-the-art proposal and with other non-transcoding-specific optimizations will also be presented at the end of this section.

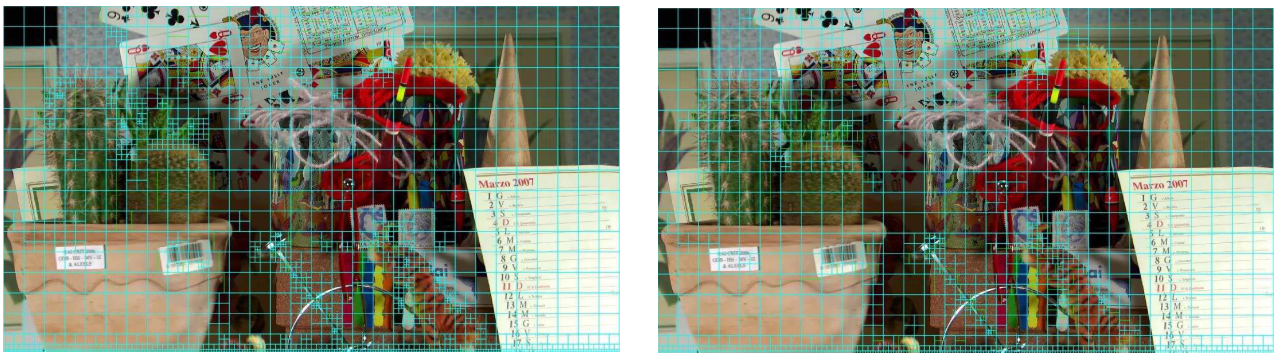


Fig. 5. CU splitting for the original and proposed algorithms. The fourth frame of the *Cactus* sequence. RA Configuration. Original splitting in the left column and splitting with the proposed algorithm in the right column.

A. Simulation Setup and Metrics

In order to ensure a common framework, the JCT-VC defined a document [31], where test conditions are set out to homogenize comparisons between experiments. Therefore, this performance evaluation has been carried out in accordance with these guidelines. Specifically, the QP values that were used are $\{22, 27, 32, 37\}$ and the configurations are RA main 10, LB main 10, and LP main 10.

The results for each sequence and the global average value are shown. Moreover, the sequences are grouped into *Classes* (A, B, C, D, and E), as described in [31], according to their resolution. It should also be noted that (according to [31]) *Class A* cannot be used with LB or LP configurations and *Class E* (videoconferencing sequences) cannot be used with the RA configuration:

1) *Class A (*2560 × 1600 *Pixels): Traffic*, *PeopleOnStreet*, *NebutaFestival*, and *SteamLocomotiveTrain* sequences;
2) *Class B (*1920 × 1800 *Pixels): Kimono*, *ParkScene*, *Cactus*, *BQTerrace*, and *BasketballDrive* sequences;
3) *Class C (*832 × 480 *Pixels): RaceHorsesC*, *BQMall*, *PartyScene*, and *BasketballDrill* sequences;
4) *Class D (*416 × 240 *Pixels): RaceHorses*, *BQSquare*, *BlowingBubbles*, and *BasketballPass* sequences;
5) *Class E (*1280 × 720 *Pixels): FourPeople*, *Johnny*, and *KristenAndSara* sequences.

The software used is JM 18.4 [41] for H.264/AVC and HM 16.2 [13] for HEVC. The remaining coding parameters not mentioned here are kept as default in the configuration file. Thus, the process to generate these results is the following:

1) Encode the YUV file with H.264/AVC reference software using HM-like configuration files that are included in the reference software JM 18.4 [41].
2) Decode each file with the decoder side of the proposed transcoder, producing a YUV' file as well as all the information needed for the proposed transcoding algorithm.
3) Encode the YUV' file with the encoder side of the original transcoder (anchor).
4) Encode the YUV' file with the encoder side of the proposed transcoder for each combination of the proposed algorithms (proposed).
5) Compare the anchor with each proposed stream in order to obtain the BD rate and the speedup.

Measurements have been performed on a six-core Intel Core i7-3930K CPU running at 3.20 GHz. The results are presented in terms of speedup and BD rate. The time reduction is also used when describing the results, which is an equivalent metric to the speedup. The speedup and time reduction are calculated as indicated in (4) and (5), respectively, where t_{anchor} is the execution time of the encoder side of the nonaccelerated reference transcoder and t_{proposed} is the execution time of the encoder side of the proposed transcoder. The YUV BD rate is the weighted average of the Y, U and V components as shown in (6) (given that the Y component is four times larger than the U and V components, since the video format is 4:2:0)

$$\text{Speed-up} = \frac{t_{\text{anchor}}}{t_{\text{proposed}}} \tag{4}$$

$$\text{Time reduction}\,(\%) = \frac{t_{\text{anchor}} - t_{\text{proposed}}}{t_{\text{anchor}}} \times 100 \tag{5}$$

$$\text{BD rate}\,(\%) = \frac{4 \times \text{BD rate}_Y + \text{BD rate}_U + \text{BD rate}_V}{6}. \tag{6}$$

B. Results and Analysis

As an initial visual evaluation, Fig. 5 shows the differences between the original CU splitting that is performed by the anchor transcoder (left column), and the CU splitting that is performed by the proposed transcoder, which makes use of the proposed AFQLD algorithm (right column). It can be seen that both CU splittings are quite similar, which demonstrates the accuracy of the classifying algorithm.

Tables III–V contain the results for RA, LB, and LP configurations, respectively. Tables III–V show the difference made by applying the AFQLD algorithm to incremental depth levels of the CU so that the evolution of the speedup and BD rate can be appreciated. Thus, it can be seen that the more levels the algorithm is applied to, the greater the speedup and the BD rate are. Moreover, the results show that Level 0 can achieve a moderate speedup without a significant loss. Consequently, the quality complexity could be adjusted by the user deciding whether to apply the AFQLD algorithm to one, two, or three levels.

It should be noted that the best performance of the proposed algorithm is not achieved for the sequences that the training frames came from. On the contrary, we have achieved the

TABLE III
SPEEDUP AND CODING EFFICIENCY RESULTS OF THE AFQLD ALGORITHM WITH RA CONFIGURATION

		BD-rate (%)												Speed-up		
		Level 0				Level 1				Level 2				L0	L1	L2
		Y	U	V	*YUV*	Y	U	V	*YUV*	Y	U	V	*YUV*			
Class A	Traffic	1.1	0.3	0.1	*0.8*	3.4	1.1	0.8	*2.6*	4.2	0.6	0.3	*3.0*	1.52	2.09	2.71
	PeopleOnStreet	0.1	-0.1	-0.1	*0.0*	0.7	0.6	0.5	*0.6*	2.0	0.7	0.6	*1.6*	1.12	1.33	1.73
	NebutaFestival	1.2	0.4	0.3	*0.9*	4.1	1.3	2.0	*3.3*	5.7	-2.0	-1.5	*3.2*	1.12	1.49	2.03
	SteamLocomotive	1.5	-0.2	-0.5	*0.9*	3.2	0.5	1.3	*2.2*	3.2	0.0	0.3	*2.2*	1.63	2.09	2.37
Class B	Kimono	0.9	0.5	-0.1	*0.7*	2.8	1.7	1.1	*2.4*	3.1	1.2	0.4	*2.3*	1.38	1.88	2.42
	ParkScene	1.2	-0.1	-0.1	*0.8*	4.3	1.1	0.5	*3.2*	4.7	0.5	0.1	*3.3*	1.45	1.95	2.52
	Cactus	0.7	0.3	0.3	*0.6*	3.1	1.6	1.5	*2.6*	3.8	1.1	1.4	*2.9*	1.42	1.90	2.39
	BasketballDrive	0.8	0.1	0.4	*0.6*	5.2	2.5	3.4	*4.4*	5.8	2.2	3.2	*4.7*	1.35	1.87	2.26
	BQTerrace	1.8	0.0	0.0	*1.2*	5.8	0.6	0.9	*4.1*	6.9	-1.3	0.7	*4.3*	1.59	2.09	2.74
Class C	BasketballDrill	0.2	0.3	0.3	*0.2*	3.1	2.6	3.0	*3.0*	3.9	3.0	3.3	*3.7*	1.25	1.62	2.10
	BQMall	0.5	0.3	0.4	*0.5*	3.0	1.2	1.8	*2.5*	3.7	1.2	1.7	*2.9*	1.27	1.69	2.22
	PartyScene	0.2	-0.3	0.1	*0.1*	1.4	0.5	0.6	*1.1*	1.9	0.1	0.2	*1.3*	1.12	1.47	1.96
	RaceHorsesC	0.1	-0.3	-0.1	*0.0*	1.3	0.7	0.7	*1.1*	1.5	0.9	1.0	*1.3*	1.11	1.45	1.46
Class D	BasketballPass	0.2	-0.2	0.4	*0.2*	4.5	1.6	3.8	*3.9*	5.1	1.8	4.6	*4.5*	1.12	1.51	1.86
	BQSquare	0.7	-0.3	-0.2	*0.4*	2.8	-0.4	-0.3	*1.7*	3.2	-1.2	-0.6	*1.8*	1.18	1.71	2.27
	BlowingBubbles	0.2	0.2	-0.1	*0.2*	1.5	0.1	0.3	*1.1*	2.1	-0.2	0.0	*1.4*	1.13	1.45	1.96
	RaceHorses	0.1	0.0	0.2	*0.1*	0.5	0.5	0.5	*0.5*	1.6	0.6	0.8	*1.3*	1.06	1.25	1.66
Average		0.7	0.1	0.1	*0.5*	3.0	1.1	1.1	*2.4*	3.7	0.5	0.9	*2.7*	1.28	1.70	2.16

TABLE IV
SPEEDUP AND CODING EFFICIENCY RESULTS OF THE AFQLD ALGORITHM WITH LB CONFIGURATION

		BD-rate (%)												Speed-up		
		Level 0				Level 1				Level 2				L0	L1	L2
		Y	U	V	*YUV*	Y	U	V	*YUV*	Y	U	V	*YUV*			
Class B	Kimono	1.0	0.1	0.6	*0.8*	3.8	0.9	1.2	*2.9*	4.3	0.7	-0.2	*2.9*	1.31	1.88	2.57
	ParkScene	1.6	-0.8	-1.7	*0.7*	7.6	-2.7	-2.9	*4.1*	8.1	-4.1	-3.9	*4.1*	1.35	1.86	2.50
	Cactus	1.7	0.0	0.1	*1.2*	5.3	2.9	2.1	*4.3*	5.9	1.3	1.5	*4.4*	1.38	1.86	2.44
	BasketballDrive	0.9	0.5	0.4	*0.8*	4.7	2.7	2.4	*4.0*	5.4	2.0	2.3	*4.3*	1.28	1.76	2.27
	BQTerrace	6.1	-3.1	-2.2	*3.2*	15.0	-2.2	-1.5	*9.4*	15.9	-8.2	-4.8	*8.4*	1.55	2.08	2.75
Class C	BasketballDrill	0.9	1.0	0.6	*0.8*	2.0	2.5	2.2	*2.1*	2.1	1.6	1.7	*2.0*	1.28	1.65	1.94
	BQMall	0.6	0.8	-1.0	*0.4*	1.8	2.2	0.6	*1.7*	2.2	1.2	0.0	*1.7*	1.26	1.59	1.99
	PartyScene	0.2	0.0	-0.9	*0.0*	0.8	0.9	1.5	*0.9*	1.3	-0.4	-0.3	*0.8*	1.15	1.42	1.83
	RaceHorsesC	0.1	-0.7	-0.3	*0.0*	0.8	-0.2	0.6	*0.6*	1.6	-0.5	0.7	*1.1*	1.16	1.35	1.74
Class D	BasketballPass	0.2	0.2	0.0	*0.2*	1.7	2.0	1.2	*1.7*	3.1	2.5	3.0	*3.0*	1.14	1.51	1.89
	BQSquare	0.3	-0.5	0.6	*0.2*	2.3	-1.5	0.9	*1.4*	5.0	-0.9	1.1	*3.4*	1.23	1.61	2.19
	BlowingBubbles	0.1	0.0	-0.8	*-0.1*	0.6	0.1	0.8	*0.6*	2.1	0.1	1.1	*1.6*	1.13	1.35	1.87
	RaceHorses	0.1	0.4	-0.3	*0.1*	0.6	1.2	1.0	*0.8*	2.2	2.0	2.6	*2.2*	1.08	1.30	1.68
Class E	FourPeople	3.9	0.3	-0.3	*2.6*	8.2	2.7	2.5	*6.3*	8.7	2.8	2.6	*6.7*	2.49	3.36	3.95
	Johnny	6.1	1.3	0.8	*4.4*	11.2	4.0	3.2	*8.7*	11.2	4.0	3.7	*8.8*	2.93	4.35	4.64
	KristenAndSara	6.9	2.2	2.6	*5.4*	11.8	5.4	5.6	*9.7*	11.9	5.1	4.9	*9.6*	2.54	3.51	4.07
Average		1.9	0.1	-0.1	*1.3*	4.9	1.3	1.3	*3.7*	5.7	0.6	1.0	*4.1*	1.52	2.03	2.52

best performance with other sequences that do not belong to the training set. This confirms that the selection of training sequences meets the requirement of being a representative sample of the common video sequences.

It can be observed that for Depth Level 0, most of the sequences achieve very good results in terms of BD-rate penalty (lower than 1% on average), with an average speedup of around 1.4× (with a time reduction of 28.6%), but some low complexity sequences in terms of SI even achieve a bigger time reduction. This proves that by applying a good classifier for just the largest CU sizes (64×64), we can achieve a moderate time reduction with a negligible rate penalty.

The proposed algorithm implementation for Depth Level 1 (i.e., Level 0 + Level 1) achieves better complexity reduction, with an average speedup of around 1.8× (Time Reduction of 45.1%) and a BD-rate penalty of around 3.1% on average. The full implementation of the algorithm for Depth Level 2 (i.e., Level 0 + Level 1 + Level 2) achieves a quantitative speedup of around 2.3× on average (with a time reduction of 56.6%) and a BD-rate penalty of around 3.4%.

In a more detailed analysis of Tables III-V, one can see some interesting results, such as obtaining negative BD rates (i.e., Level 0) or obtaining lower BD rates when adding one more level of depth (i.e., *BQTerrace* sequence when using the LB configuration). This happens because both ME and AFQLD algorithms are optimized for the luminance component, but not for chrominance. In fact, the BD rate of the luminance is worse when adding depth levels to the algorithm, but the BD rate of the chrominances obtains better results and, after weighting the components, the global BD rate is better. It can also be seen that the LB configuration obtains a higher speedup than RA, on average. However, this is due to the fact that Class E obtains very high speedups (as it contains videoconferencing sequences, which are easy to predict due

TABLE V
SPEEDUP AND CODING EFFICIENCY RESULTS OF THE AFQLD ALGORITHM WITH LP CONFIGURATION

		BD-rate (%)												Speed-up		
		Level 0				Level 1				Level 2				L0	L1	L2
		Y	U	V	*YUV*	Y	U	V	*YUV*	Y	U	V	*YUV*			
Class B	Kimono	0.4	-0.1	0.8	*0.4*	2.6	1.8	3.0	*2.5*	3.0	1.3	1.6	*2.5*	1.21	1.64	2.30
	ParkScene	0.4	1.2	0.1	*0.5*	5.7	2.2	2.4	*4.6*	6.4	0.4	0.2	*4.3*	1.24	1.74	2.29
	Cactus	0.3	-0.2	0.2	*0.2*	2.8	3.7	1.6	*2.8*	4.2	0.1	0.5	*2.9*	1.27	1.53	2.02
	BasketballDrive	0.1	0.3	0.2	*0.2*	3.6	3.1	2.7	*3.4*	4.5	2.4	2.4	*3.8*	1.16	1.58	2.00
	BQTerrace	0.4	0.8	-1.4	*0.2*	9.5	2.2	3.3	*7.3*	10.6	-3.5	-0.9	*6.3*	1.31	1.81	2.28
Class C	BasketballDrill	0.0	-0.1	0.1	*0.0*	1.7	1.5	2.0	*1.7*	3.3	2.0	3.1	*3.0*	1.16	1.45	1.91
	BQMall	0.2	-0.5	-0.2	*0.0*	1.1	0.9	1.0	*1.1*	2.4	0.0	-0.2	*1.6*	1.13	1.33	1.87
	PartyScene	0.0	0.8	0.1	*0.2*	0.5	2.0	0.9	*0.8*	1.5	0.8	-0.7	*1.0*	1.08	1.26	1.67
	RaceHorsesC	0.0	-0.1	-0.6	*-0.1*	0.6	0.9	0.4	*0.6*	1.3	1.1	0.8	*1.2*	1.08	1.26	1.59
Class D	BasketballPass	0.0	0.0	-0.2	*0.0*	1.3	0.5	1.0	*1.1*	2.2	1.4	1.6	*2.0*	1.09	1.33	1.65
	BQSquare	0.1	0.6	-2.1	*-0.2*	2.3	-1.9	-0.8	*1.1*	3.0	-0.4	-3.2	*1.4*	1.08	1.43	1.79
	BlowingBubbles	0.0	0.0	1.2	*0.2*	0.3	1.3	2.3	*0.8*	1.5	-0.3	0.2	*1.0*	1.06	1.22	1.64
	RaceHorses	0.1	0.1	0.0	*0.1*	0.5	0.3	0.6	*0.5*	1.3	0.7	0.8	*1.1*	1.06	1.23	1.51
Class E	FourPeople	1.7	-0.9	-0.6	*0.9*	7.2	2.3	2.9	*5.6*	7.5	1.8	2.0	*5.6*	1.94	2.57	3.36
	Johnny	5.6	2.2	1.9	*4.4*	13.4	4.7	3.3	*10.2*	11.9	3.9	2.0	*8.9*	1.64	3.50	4.30
	KristenAndSara	3.7	2.2	2.0	*3.2*	9.4	5.4	5.1	*8.0*	10.0	6.1	5.2	*8.6*	2.13	2.79	3.66
Average		0.8	0.4	0.1	*0.6*	3.9	1.9	2.0	*3.3*	4.7	1.1	1.0	*3.5*	1.29	1.73	2.24

TABLE VI
ACTUAL HIT RATE FOR EACH LEVEL OF THE PROPOSED AFQLD ALGORITHM

		Hit rates for RA configuration (%)			Hit rates for LB configuration (%)			Hit rates for LP configuration (%)		
		Level 0	Level 1	Level 2	Level 0	Level 1	Level 2	Level 0	Level 1	Level 2
Class A	Traffic	90.93	83.34	92.75	NA	NA	NA	NA	NA	NA
	PeopleOnStreet	96.12	88.02	88.67	NA	NA	NA	NA	NA	NA
	NebutaFestival	91.79	80.88	89.23	NA	NA	NA	NA	NA	NA
	SteamLocomotive	86.73	85.28	93.87	NA	NA	NA	NA	NA	NA
Class B	Kimono	89.26	83.84	95.78	85.34	79.73	94.09	87.93	78.53	93.73
	ParkScene	90.81	87.01	93.96	88.57	79.06	90.31	90.86	78.12	88.46
	Cactus	93.38	85.85	94.01	91.58	80.69	90.32	92.86	82.76	86.50
	BQTerrace	91.75	87.79	94.53	87.51	82.49	91.50	90.34	81.89	87.54
	BasketballDrive	89.77	83.14	92.85	86.41	79.52	89.71	91.08	84.09	92.59
Class C	RaceHorsesC	96.44	90.17	91.87	94.55	85.04	86.63	94.86	85.25	82.85
	BQMall	93.09	86.18	90.58	91.71	83.33	86.40	93.24	83.47	82.80
	PartyScene	94.48	88.64	91.39	93.61	84.21	87.29	95.25	86.09	82.59
	BasketballDrill	94.54	84.82	90.22	91.20	82.92	85.90	92.81	82.96	84.46
Class D	RaceHorses	97.13	91.32	91.86	97.68	89.44	84.70	98.08	90.30	84.56
	BQSquare	92.86	91.76	92.85	91.96	86.86	85.65	96.01	89.68	83.64
	BlowingBubbles	96.83	93.14	93.50	94.60	88.06	83.52	96.28	90.77	82.53
	BasketballPass	95.94	89.83	92.06	95.21	88.79	85.45	96.65	90.76	85.79
Class E	FourPeople	NA	NA	NA	91.25	81.47	92.43	94.89	82.63	90.48
	Johnny	NA	NA	NA	91.67	89.51	94.46	93.43	83.62	96.04
	KristenAndSara	NA	NA	NA	90.42	85.63	95.00	92.21	84.18	93.58
Average		93.09	87.12	92.35	91.45	84.17	88.96	93.55	84.69	87.38

to static backgrounds and few details), but when looking at the rest of the sequences in Tables III and IV, it can be seen that RA obtains better results than LB since the ME module is more complex for the RA configuration. Furthermore, it can be seen in Table IV that Class E has the worst BD rate, although this is due to the fact that videoconferencing achieves very good coding performance in the anchor transcoder, so increasing the bitrate in the same absolute quantity as in other classes (or even in a lower quantity) results in a higher relative increase, which is what the BD-rate measures. All these facts are also present in the LP configuration.

Finally, the results obtained prove a good tradeoff between the complexity reduction and the encoding performance provided by our proposal, which can be applied in a scalable way. The global experimental results confirm that the proposed algorithm can reduce the computational complexity of the H.264/HEVC transcoder by more than a half with a slight BD-rate increase, favoring the real-time software and hardware implementation.

C. Actual Hit Rate of Classifying Models

Another way to evaluate the algorithm is by looking at the actual hit rate (that is, the rate of correctly classified CU splittings) achieved by the algorithm. Table VI shows these hit rates. The hit rate for a sequence is shown as the average hit rate for the four QP values described above.

It can be seen that high hit rates are achieved on average, which demonstrates the accuracy of the classifying algorithm, which cannot be easily improved upon, since hit rates higher

TABLE VII
COMPARISON BETWEEN PTCM-LDF, FQLD, AND AFQLD ALGORITHMS USING LP CONFIGURATION WITH ONE REFERENCE FRAME

		AFQLD		PTCM-LDF [21]		FQLD [29]	
		BD-Rate (%)	Speed-up	BD-Rate (%)	Speed-up	BD-Rate (%)	Speed-up
Class B	Kimono	2.3	2.30	2.6	2.29	2.3	2.73
	ParkScene	1.8	2.22	3.7	2.68	0.9	2.53
	Cactus	3.4	2.03	5.0	2.38	6.1	2.20
	BasketballDrive	3.2	1.99	4.0	1.81	7.6	2.22
Class C	BasketballDrill	2.7	1.74	5.7	1.78	4.3	2.09
	BQMall	1.6	1.69	5.9	2.09	4.0	2.10
	PartyScene	0.9	1.52	3.4	1.96	2.2	2.04
	RaceHorsesC	0.9	1.50	5.1	1.96	3.7	1.94
Average		2.1	1.87	4.4	2.12	3.9	2.23

TABLE VIII
COMPARISON BETWEEN AFQLD, ECU, AFQLD + ESD + CFM, AND ECU + ESD + CFM

		AFQLD		ECU		AFQLD+ESD+CFM		ECU+ESD+CFM	
		BD-Rate (%)	Speed-up	BD-Rate (%)	Speed-up	BD-Rate (%)	Speed-up	BD-Rate (%)	Speed-up
Class A	Traffic	3.0	2.71	1.4	2.05	4.4	3.72	3.8	2.79
	PeopleOnStreet	1.6	1.73	0.9	1.24	2.6	2.08	2.8	1.52
	NebutaFestival	3.2	2.03	8.0	1.32	6.7	2.30	15.9	1.52
	SteamLocomotive	2.2	2.37	1.0	1.32	3.2	3.09	2.5	2.35
Class B	Kimono	2.3	2.42	1.2	1.74	3.4	3.08	3.0	2.14
	ParkScene	3.3	2.52	2.1	1.86	5.0	3.27	5.4	2.36
	Cactus	2.9	2.39	2.1	1.79	4.5	3.05	5.2	2.26
	BasketballDrive	4.7	2.26	1.3	1.62	6.3	2.78	3.7	2.00
	BQTerrace	4.3	2.74	4.4	2.05	7.4	3.68	10.5	2.69
Class C	BasketballDrill	3.7	2.10	1.6	1.52	5.3	2.59	4.4	1.87
	BQMall	2.9	2.22	1.5	1.68	4.1	2.86	4.2	2.14
	PartyScene	1.3	1.96	3.0	1.53	2.7	2.44	6.8	1.89
	RaceHorsesC	1.3	1.46	1.5	1.27	3.4	2.09	4.9	1.51
Class D	BasketballPass	4.5	1.86	1.7	1.34	6.4	2.22	4.8	1.63
	BQSquare	1.8	2.27	2.4	1.75	4.0	3.02	7.0	2.42
	BlowingBubbles	1.4	1.96	2.9	1.57	2.7	2.47	6.3	1.96
	RaceHorses	1.3	1.66	1.8	1.22	3.2	1.96	5.4	1.47
Average		2.7	2.16	2.3	1.61	4.4	2.75	5.7	2.03

than 95% are not usually achieved in real applications (and some of them are even higher than that value). In fact, it can be seen that all values are higher than 80%. In the case of Class E with LB and LP configurations, the values range from 81% to 96%, which strengthens the idea that the classifiers work equally well for Class E as for other classes even though the BD rate is higher (due to the good performance of the original encoder for these sequences in terms of RD, as stated above).

D. Comparison With Other Transcoding Optimizations

Table VII shows the comparison between the AFQLD, PTCM-LDF [21], and FQLD [29] algorithms. It should be noted that PTCM-LDF can be parametrized with the number of frames used to train and the length of the sequence. The shown results correspond to the average value of sequences of lengths of 2.5, 5, and 10 s and using 10 frames for the training (this way of showing the results is the same as Peixoto *et al.* [21] used to summarize them).

In this case, the same configuration as in [21] (LP configuration with only onereference frame) is used for AFQLD and FQLD in order to obtain comparable results to those presented. The results show the BD rate and the speedup for all the sequences that were used in [21]. It can be clearly seen that AFQLD outperforms both PTCM-LDF and FQLD in terms of BD rate. In the case of speedup, AFQLD results are a bit lower, but it should be noted that in terms of time reduction, AFQLD obtains 47% and FQLD obtains 55%, which are really close values, and the wide gain in BD rate makes AFQLD to perform better globally. This same fact happens when the comparison is between AFQLD and PTCM-LDF, but in this case, the speedups are even closer, since the time reduction of PTCM-LDF is 52%.

E. Comparison With Non-Transcoding-Specific Optimizations

Table VIII shows the comparison between the AFQLD algorithm and the nontranscoding-specific optimizations integrated in HM reference software, namely, ECU [16], ESD [14], and CFM [15], and described in Section II, using RA configuration. The idea is to compare our proposal (using level $=$ 2) with respect to another transcoder where the re-encoding stage enables the aforementioned techniques. A statistical comparison following the methodology presented in [42] has been carried out using data from Table VIII. First of all, a Friedman test for the BD-rates and the speedups has been used, with the result that all methods are not equivalent (in both, BD-rate and speedup terms). Then, a posthoc test using an Holm adjust method has been used in order to compare pairwise methods. These tests show that ECU

and AFQLD algorithms present comparable BD-rate values, while AFQLD outperforms ECU in terms of speedup. This result is also true when comparing AFQLD + ESD + CFM and ECU + ESD + CFM. Moreover, when comparing AFQLD and ECU + ESD + CFM, the tests show that the speedups are comparable, while AFQLD outperforms ECU + ESD + CFM in BD rate, showing the effectiveness of the proposed algorithm.

VI. Conclusion

In this paper, we have presented a new approach for accelerating the CU partitioning decision of an H.264/HEVC video transcoder, based on a statistical NB classifier algorithm, which decides on the most appropriate quadtree level by taking into account what has occurred in the H.264/AVC decoder, without the need for testing all the CUs/PUs. This allows an early classification of CUs/PUs, avoiding the exhaustive RDO evaluation of all the available CU sizes. Moreover, this algorithm is adaptive due to the fact that the algorithm can be dynamically adapted to the content of each sequence.

The simulation results obtained demonstrate a good tradeoff between the complexity reduction and the encoding performance provided by our proposal, which can be applied in an incremental way. The full implementation of the AFQLD algorithm for the three levels achieves a quantitative speedup of around 2.31× on average (with a time reduction of 56.7%) and a BD-rate penalty of around 3.4%, compared with the anchor transcoder, for a wide range of resolutions (from Classes A to E). For moderate complexity reduction, our AFQLD algorithm also allows the implementation of only the first depth level, obtaining a 26.7% time saving with a negligible rate penalty of around 0.8%. Furthermore, the AFQLD algorithm can achieve better BD rates than a state-of-the-art transcoding algorithm, such as PTCM-LDF [21] and FQLD [29]. It can also obtain better performance than non-transcoding specific algorithms, such as ECU, ESD, or CFM.

Finally, the proposed algorithm is suitable for H.264/HEVC hardware and software real-time transcoder realization, thanks to the noncomplex classifier implemented and a set of noncomplex computable features.

Acknowledgment

The authors would like to thank E. Peixoto for sharing the results of his experiments.

References

[1] *High Efficiency Video Coding*, document ITU-T Rec. H.265 and ISO/IEC 23008-2 (Version 1), Apr. 2013.

[2] *High Efficiency Video Coding*, document ITU-T Rec. H.265 and ISO/IEC 23008-2 (Version 2), Oct. 2014.

[3] *Advanced Video Coding for Generic Audiovisual Services*, document ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) (Version 22), Feb. 2012.

[4] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—Including High Efficiency Video Coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, Dec. 2012.

[5] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[6] C. C. Chi *et al.*, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1827–1838, Dec. 2012.

[7] G. Bjøntegaard, *Improvements of the BD-PSNR Model*, ITU-T SG16 Q6, document VCEG-AI11, Jul. 2008.

[8] W.-J. Han *et al.*, "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1709–1720, Dec. 2010.

[9] L. W. K. Panusopone and X. Fang, *Efficient Transform Unit Representation*, document JCTVC-D250, Jan. 2011.

[10] X. Li, M. Wien, and J.-R. Ohm, "Rate-complexity-distortion optimization for hybrid video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 7, pp. 957–970, Jul. 2011.

[11] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Performance and computational complexity assessment of High-Efficiency Video Encoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1899–1909, Dec. 2012.

[12] F. Bossen, B. Bross, K. Sühring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec. 2012.

[13] *HM Reference Software*. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/, accessed Mar. 1, 2015.

[14] J. Yang, J. Kim, K. Won, H. Lee, and B. Jeon, *Early SKIP Detection for HEVC*, document JCTVC-G543, Nov. 2011.

[15] R. H. Gweon and Y.-L. Lee, *Early Termination of CU Encoding to Reduce HEVC Complexity*, document JCTVC-F045, Jul. 2011.

[16] K. Choi, S. Park, and E. S. Jang, *Coding Tree Pruning Base CU Early Termination*, document JCTVC-F902, Jul. 2011.

[17] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 18–29, Mar. 2003.

[18] D. Zhang, B. Li, J. Xu, and H. Li, "Fast transcoding from H.264 AVC to High Efficiency Video Coding," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Melbourne, Vic., Australia, Jul. 2012, pp. 651–656.

[19] E. Peixoto and E. Izquierdo, "A complexity-scalable transcoder from H.264/AVC to the new HEVC codec," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Orlando, FL, USA, Sep. 2012, pp. 737–740.

[20] E. Peixoto, B. Macchiavello, E. M. Hung, A. Zaghetto, T. Shanableh, and E. Izquierdo, "An H.264/AVC to HEVC video transcoder based on mode mapping," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Melbourne, Vic., Australia, Sep. 2013, pp. 1972–1976.

[21] E. Peixoto, T. Shanableh, and E. Izquierdo, "H.264/AVC to HEVC video transcoder based on dynamic thresholding and content modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 99–112, Jan. 2014.

[22] E. Peixoto, B. Macchiavello, E. M. Hung, and R. L. de Queiroz, "A fast HEVC transcoder based on content modeling and early termination," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 2532–2536.

[23] P. Xing, Y. Tian, X. Zhang, Y. Wang, and T. Huang, "A coding unit classification based AVC-to-HEVC transcoding with background modeling for surveillance videos," in *Proc. Vis. Commun. Image Process. (VCIP)*, Nov. 2013, pp. 1–6.

[24] W. Jiang, Y. Chen, and X. Tian, "Fast transcoding from H.264 to HEVC based on region feature analysis," *Multimedia Tools Appl.*, vol. 73, no. 3, pp. 2179–2200, Dec. 2014.

[25] L. P. Van *et al.*, "Fast transrating for High Efficiency Video Coding based on machine learning," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Melbourne, Vic., Australia, Sep. 2013, pp. 1573–1577.

[26] J. De Praeter, J. De Cock, G. Van Wallendael, S. Van Leuven, P. Lambert, and R. Van de Walle, "Efficient transcoding for spatially misaligned compositions for HEVC," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 2487–2491.

[27] T. Shanableh, E. Peixoto, and E. Izquierdo, "MPEG-2 to HEVC video transcoding with content-based modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 7, pp. 1191–1196, Jul. 2013.

[28] A. J. Diaz-Honrubia, J. L. Martinez, and P. Cuenca, "Multiple reference frame transcoding from H.264/AVC to HEVC," in *Proc. Int. Conf. Multimedia Model. (MMM)*, Dublin, Ireland, Jan. 2014, pp. 593–604.

[29] A. J. Diaz-Honrubia, J. L. Martinez, J. M. Puerta, J. A. Gamez, J. de Cock, and P. Cuenca, "Fast quadtree level decision algorithm for H.264/HEVC transcoder," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 2497–2501.

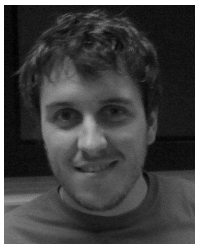
[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[31] F. Bossen, *Common HM Test Conditions and Software Reference Configurations*, document JCTVC-L1100, Jan. 2013.

[32] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," *Commun. ACM*, vol. 39, no. 11, pp. 27–34, Nov. 1996.
[33] G. Fernandez-Escribano, H. Kalva, P. Cuenca, L. Orozco-Barbosa, and A. Garrido, "A fast MB mode decision algorithm for MPEG-2 to H.264 P-frame transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, pp. 172–185, Feb. 2008.
[34] S. Patnaik and Y.-M. Yang, *Soft Computing Techniques in Vision Science*, vol. 395. New York, NY, USA: Springer-Verlag, 2012.
[35] *Subjective Video Quality Assessment Methods for Multimedia Applications*, document ITU-R P.910, 1999.
[36] U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proc. Int. Joint Conf. Uncertainty AI*, Aug. 1993, pp. 1022–1027.
[37] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.
[38] M. Minsky, "Steps toward artificial intelligence," *Trans. Inst. Radio Eng.*, vol. 4, pp. 8–30, Jan. 1961.
[39] P. Domingos and M. Pazzani, "Beyond independence: Conditions for the optimality of the simple Bayesian classifier," in *Proc. Int. Conf. Mach. Learn.*, Jul. 1996, pp. 105–112.
[40] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques* (The Morgan Kaufmann Series in Data Management Systems). San Mateo, CA, USA: Morgan Kaufmann, 2011.
[41] *JM Reference Software, Version 18.4*. [Online]. Available: http://iphome.hhi.de/suehring/tml/, accessed May 14, 2014.
[42] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

Antonio Jesús Díaz-Honrubia (S'14) received the bachelor's degree in computer science and engineering, and the M.Sc. degrees in advanced computing technologies and computer science and engineering, from University of Castilla-La Mancha, Albacete, Spain, in 2011 and 2012, respectively, where he is currently working toward the Ph.D. degree in advanced computing technologies.

He is with University of Castilla-La Mancha, where he carries out his research activities with the Computer Architecture and Technology Group, Albacete Research Institute of Informatics. His research interests include video transcoding from H.264/AVC to the High Efficiency Video Coding standard and 3D video coding.



José Luis Martínez received the M.Sc. and Ph.D. degrees in computer science and engineering from University of Castilla-La Mancha, Albacete, Spain, in 2005 and 2009, respectively.

He was a Post-Doctoral Researcher with the Centre for Communication System Research, University of Surrey, Guildford, U.K. In 2011, he joined University of Castilla-La Mancha, where he is currently an Assistant Professor. He was a Visiting Researcher with Florida Atlantic University, Boca Raton, FL, USA, for nine months. He has 26 publications in these areas in international refereed journals and 60 conference proceedings. His research interests include video transcoding, scalable video coding, and video applications for multicore and GPU architectures.

Pedro Cuenca received the M.Sc. (Hons.) degree in physics with a minor in electronics and computer science from University of Valencia, Valencia, Spain, in 1994, and the Ph.D. degree in computer engineering from Polytechnic University of Valencia, Valencia, in 1999.

He joined the Department of Computer Engineering, University of Castilla-La Mancha, Albacete, Spain, in 1995, where he is currently a Full Professor. He has been a Visiting Researcher with Nottingham Trent University, Nottingham, U.K.; University of Ottawa, Ottawa, ON, Canada; and University of Surrey, Guildford, U.K. He has authored over 150 papers in international journals and conferences. His research interests include video compression, QoS video transmission, and video applications for multicore and GPU architectures.

José Antonio Gamez received the M.S. degree in computer science and the Ph.D. degree in computer science from University of Granada, Granada, Spain, in 1991 and 1998, respectively.

He joined the Department of Computer Science, University of Castilla-La Mancha, Albacete, Spain, in 1991, where he is currently a Full Professor. He has edited five books and five special issues of international journals. He has (co-)authored over 100 papers in journals, books, and refereed international conferences. His research interests include probabilistic reasoning, Bayesian networks, evolutionary algorithms, machine learning, and data mining.

José Miguel Puerta received the M.S. and Ph.D. degrees in computer science from University of Granada, Granada, Spain, in 1991 and 2001, respectively.

He joined the Department of Computer Systems, University of Castilla-La Mancha, Albacete, Spain, in 1991, where he is currently an Associate Professor. He has authored over 50 papers. His research interests include probabilistic graphical models, Bayesian networks, evolutionary algorithms, machine learning, and data mining.