Probabilistic Approach Versus Machine Learning for One-Shot Quad-Tree Prediction in an Intra HEVC Encoder

Alexandre Mercat, Florian Arrestier, Maxime Pelcat, Wassim Hamidouche, Daniel Menard

Journal of Signal Processing Systems (2019) 91:1021–1037
https://doi.org/10.1007/s11265-018-1426-z

Probabilistic Approach Versus Machine Learning for One-Shot Quad-Tree Prediction in an Intra HEVC Encoder

Alexandre Mercat[1] · **Florian Arrestier**[1] · **Maxime Pelcat**[1,2] · **Wassim Hamidouche**[1] · **Daniel Menard**[1]

Received: 2 March 2018 / Revised: 26 September 2018 / Accepted: 23 November 2018 / Published online: 14 December 2018

Abstract

Evolutions of the Internet of Things (IoT) in the next years are likely to boost mobile video demand to an unprecedented level. A large number of battery-powered systems will integrate an HEVC video codec, implementing the latest encoding MPEG standard, and these systems will need to be energy efficient. Constraining the energy consumption of HEVC encoders is a challenging task, especially for embedded applications based on software encoders. The most efficient approach to reduce the energy consumption of an HEVC encoder consists in optimizing the quad-tree block partitioning of the image and trade-off compression efficiency and energy consumption by efficiently choosing the near-optimal pixel block sizes. For the purpose of reducing the energy consumption of a real-time HEVC Intra encoder, this paper proposes and compares two methods that predict the quad-tree partitioning in "one-shot", i.e. without iterating. These methods drastically limit the computational cost of the recursive Rate-Distortion Optimization (RDO) process. The first proposed method uses a *Probabilistic* approach whereas the second method is based on *Machine Learning* approach. Experimental results show that both methods are capable of reducing the energy consumption of an embedded HEVC encoder of 58% for a bit rate increase of respectively 3.93% and 3.6%.

Keywords HEVC · Intra · Quad-tree prediction · One-shot · Machine learning · Energy · Real-time

1 Introduction

With the progress of microelectronics, many embedded applications now encode and stream live video contents. The HEVC [33, 34, 42] standard represents the state-of-the-art video standard. When compared with the previous ISO/IEC Moving Picture Experts Group (MPEG) Advanced Video Coding (AVC) standard, HEVC Main profile reduces the bit rate by 50% on average for a similar objective video quality [35, 37]. This gain reduces the energy needed for transmitting video. On the other hand, the computational complexity of the encoders has been significantly increased. The additional complexity brought by HEVC is mostly due to the new quad-tree block partitioning structure of Coding Tree Units (CTUs) and the increase in the number of Intra prediction modes, which exponentially impact the Rate-Distortion Optimization (RDO) process [20].

The main limitation of recent embedded systems, particularly in terms of computational performance, comes from the bounded energy density of batteries. This limitation is a major constraint for image and video applications, video encoding and decoding being for instance the most energy-consuming algorithms on smart phones [3]. A large share of systems are likely to integrate the HEVC codec in the long run and will require to be energy efficient, and even energy aware. As a consequence, energy consumption represents a serious challenge for embedded HEVC real-time encoders. For both hardware and software

✉ Alexandre Mercat
alexandre.mercat@insa-rennes.fr

Florian Arrestier
florian.arrestier@insa-rennes.fr

Maxime Pelcat
maxime.pelcat@insa-rennes.fr

Wassim Hamidouche
wassim.hamidouche@insa-rennes.fr

Daniel Menard
daniel.menard@insa-rennes.fr

[1] Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, 20 Avenue des Buttes de Coesmes, 35708 Rennes, France

[2] Institut Pascal, CNRS UMR 6602, Clermont-Ferrand, France

codecs, a solution to reduce energy consumption is to decrease the computational complexity while controlling compression quality losses.

To reduce the computational complexity of HEVC encoders, several algorithmic solutions have been proposed at the level of quad-tree partitioning. Indeed, choosing the right encoding block sizes is necessary to obtain a good compression ratio but this choice is difficult and usually results from a costly RDO process. The exhaustive search partitioning solution is the optimal one, obtained by testing all possible partitioning configurations and selecting the one that minimizes the Rate-Distortion (RD)-cost. This process is the most time consuming operation in an HEVC encoder and thus it offers the biggest opportunity of complexity reduction (up to 78% in the considered embedded encoder) [20]. Complexity reduction solutions at the quad-tree level consist in predicting, without encoding, the adequate level of partitioning that offers the lowest RD–cost.

As examples of related works, authors in [31] and [4] propose to use the correlation between the minimum depth of the co-located CTUs in the current and previous frames to skip computing some depth levels during the RDO process. Authors in [1, 9, 15, 25, 40] use CTU texture complexities to predict the quad-tree partitioning. All these solutions are based on reducing the complexity of an offline (i.e. non-real-time) costly reference encoder called HEVC test Model (HM). In this paper, we target energy reduction in a real-time context of an optimized software encoder. A real-time encoder such as Kvazaar is up to 10 times faster than HM [39]. The complexity reduction performance of state-of-the-art solutions based on HM are biased since they are measured with respect to a large compression time. The complexity overhead of state-of-the-art solutions is thus comparatively higher in the context of a real-time encoder.

We propose in this paper two energy reduction methods for HEVC Intra encoders based on a CTU partitioning prediction technique. We then compare these methods that drastically limit the recursive RDO process. The first method exploits the correlation between CTU partitioning and the variance of the CTU luminance samples to predict the quad-tree decomposition in one-shot. The second method uses a Machine Learning approach to perform the same prediction. Machine Learning is an interdisciplinary subfield of computer science that aims to replace the manually engineered solutions for extracting information from sensored data in all application fields. The two methods are compared in terms of prediction accuracy of the quad-tree partitioning as well as in terms of compression performance.

The rest of this paper is organized as follows. Section 2 presents an overview of the HEVC intra encoder and goes through State-of-the-Art methods of complexity reduction techniques. Section 3 details the first proposed probabilistic algorithm of quad-tree partitioning prediction based on variance studies. Section 4 presents the second proposed Machine Learning algorithm of quad-tree partitioning prediction. The two proposed energy reduction schemes are then compared in term of quad-tree partitioning accuracy and performance in Section 5. Finally, Section 6 concludes the paper.

2 Related works

2.1 HEVC Encoding and its Rate Distortion Optimisation

An HEVC encoder is based on a classical *hybrid video encoder* structure that combines Intra-image and Inter-images predictions. While encoding in HEVC, each frame is split into equally-sized Coding Tree Units (CTUs) (Fig. 1). Each CTU is then divided into Coding Units (CUs), appearing as nodes in a quad-tree. CUs gathers the coding information of a block of luminance and 2 blocks of chrominance (in 420 representation). In HEVC, the size, in luminance pixels, of CUs is equal to $2N \times 2N$ with $N \in \{32, 16, 8, 4\}$. The HEVC encoder first predicts the units from their neighbourhood (in space and time). To perform the prediction, CUs may be split into Prediction Units (PUs) of smaller size. In intra prediction mode, PUs are square and have a luminance size of $2N \times 2N$ (or $N \times N$ only when $N = 4$), which can be associated to a quad-tree depth range $d \in \{0, 1, 2, 3, 4\}$, as illustrated in Fig. 1.

The HEVC intra-frame prediction is complex and supports in total $N_{pm} = 35$ modes performed at the level of PU including planar (surface fitting) mode, DC (flat) mode and 33 angular modes [33]. Each mode corresponds to a different assumption on the gradient in the image. To achieve the best RD performance, the encoder performs an exhaustive search process, named Rate-Distortion Optimization (RDO), testing all possible combinations of quad-tree partitioning and the 35 Intra prediction modes. The Quantization Parameter (QP) impacts the RDO process to tune quality and bitrate. For a given CTU, an RDO exhaustive search tests N_t different decompositions and prediction modes where:

$$N_t = N_{pm} \times \sum_{d=0}^{4} 2^{2d} = 35 \times (1+4+16+64+256) = 11935 \quad (1)$$

This set of tests is the main cause of the HEVC encoding complexity and the target of the energy optimization process developed in this paper.

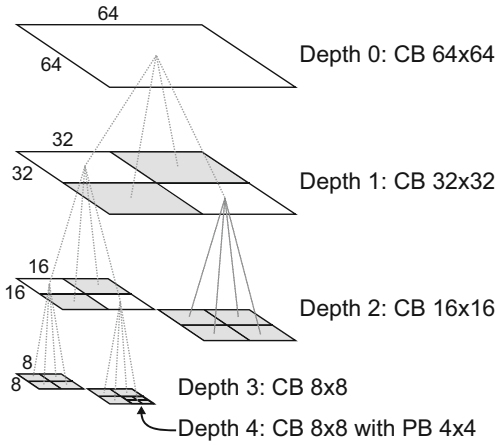


Figure 1 Quad-tree structure of a Coding Tree Unit (CTU), divided into Coding Units (CUs) and Prediction Units (PUs) (dimensions in luminance pixels).

2.2 Software Real-Time HEVC Encoder

For embedded applications, hardware encoding solutions [27] consume much lower energy than software solutions. However, when the considered system does not embed a hardware coprocessor, a software HEVC encoder [13, 24, 36, 38] can be used, for instance the HEVC reference software model (HM). HM is widely used, as it has been designed to achieve an optimal coding efficiency (in terms of RD). However, the computational complexity of HM is high and not adapted to embedded applications. To fill this gap, the *x265* [24], *f265* [38] and *Kvaazar* [36] HEVC software encoders provide real-time encoding solutions, leveraging on parallel processing and low-level Single Instruction Multiple Data (SIMD) optimizations for different specific platforms.

This study is based on the Kvaazar HEVC encoder [36] for its real-time encoding capacity of Ultra High Density (UHD) videos. The conclusions of this study can however be extended to other real time software or hardware encoders, as they all depend on a classical RDO process to reach high compression performance.

2.3 Complexity Reduction of the Quad-Tree Partitioning

As shown in [20], in a real-time software HEVC Intra encoder, two specific parts of the encoding algorithm provide the highest opportunities of energy reduction; the Intra prediction (IP) level offers at best 30% of energy reduction whereas the CTU quad-tree partitioning level has a potential of energy reduction of up to 78%. Previous studies on low complexity CTU quad-tree partitioning can be classified into two categories: the early termination complexity reduction techniques which are applied during the RDO process to dynamically terminate the process when further gains are unlikely, and the prediction-based complexity reduction techniques which are applied before starting the RDO process and predict the quad-tree partitioning with lower complexity processing. In this paper, we focus on prediction-based complexity reduction techniques.

Authors of [4, 31, 44] propose to reduce the complexity of the HEVC encoder by skipping some depth levels of the quad-tree partitioning. The skipped depths are selected based of the correlation between the minimum depth of the co-located CTUs in the current and previous frames. Results in [4] show an average time savings of 45% for a Bjøntegaard Delta Bit Rate (BD-BR) increase of 1.9%. For the algorithm from [31], results show an average complexity reduction of 21%. Concerning [44], experimental results show that the method can save about 48% encoding time for a BD-BR increase of 2.9%. In this paper, the objective of the study is to demonstrate a drastic energy reduction in a real-time encoding setup by predicting the CTU partitioning. As a consequence, higher energy reductions are obtained at the expense of higher BD-BR increases.

Works in [1, 9, 15, 25, 40] use CTU texture complexities to predict the quad-tree partitioning. Min et al. [1] propose to decide if a CU has to be split, non-split or if it is undetermined, using the global and local edge complexities in four different directions (horizontal, vertical, 45° and 135° diagonals) of CU and sub-CUs. This method provides a computational complexity reduction of 52% (in the non-real-time HM) for a BD-BR increase of 0.8%. Feng et al. [9] use information entropy of CUs and sub-CUs saliency maps to predict the CUs size. This method reduces the complexity by 37.9% (in HM) for a BD-BR increase of 0.62%.

Khan et al. [15] propose a method using texture variance to efficiently predict the CTU quad-tree decomposition. The authors model the Probability Density Function (PDF) of variance populations by a Rayleigh distribution to estimate some variance thresholds and determine the quad-tree partitioning. This method reduces the complexity by 44% (in HM) with a BD-BR increase of 1.27%. Our experiments have shown that the assumption of a Rayleigh distribution is not verified in many cases. For this reason, our Probabilistic proposed method, based on the variance, does not consider the Rayleigh distribution and thus differs from [15].

In [26], Penny et al. propose the first Pareto-based energy controller for an HEVC encoder. From [26] are extracted the following results which are the average results on one sequence of each video class (A, B, C, D et E). For an energy reduction from 49% to 71% (in HM), authors achieve a BD-BR increase between 6.84% and 25%, respectively.

Several works have been proposed that use Machine Learning based optimization to reduce the complexity of the HEVC encoding process. Authors of [29, 30] present

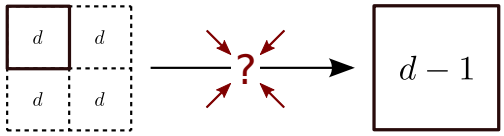


Figure 2 Merge prediction of a CU of depth d (belonging to a group of 4 neighboring CUs in the Z-scan order) into a CU of depth $d-1$.

an Intra CU size classifier based on data-mining with an offline classifier training. The classifier is a three-node decision tree using mean and variance of CUs and sub-CUs as characteristics. This algorithm reduces the coding time by 52% (in HM) at the expense of BD-BR increase of 2%. Duanmu et al. [7] present a fast CU partitioning using Machine Learning for screen content video compression. Authors use several characteristics such as CU luma variances, color Kurtosis of CU, gradient Kurtosis of CU. Shen and Yu [32] propose a CU splitting early termination algorithm based on a Support Vector Machine (SVM). The RD cost losses due to the misclassification are used as features (weights) during the SVM training. In [43], authors model the coding tree determination in HEVC with a three-level hierarchical decision problem using SVM predictors.

These studies are all based on complexity reduction of the HM software encoder and their performance can not be directly translated to real-time encoders. The two methods proposed in the next sections are studied within a real-time optimized encoder and demonstrate high prediction efficiency.

3 Probabilistic Approach for Predicting an HEVC Quad-Tree Partitioning

The aim of the techniques proposed in this paper is to replace the brute force scheme usually employed in HEVC encoders by a low complexity algorithm that predicts in one-shot the CTU partitioning for Intra prediction without testing all possible decompositions. Following a bottom-up approach (from CU 4×4 to 32×32), the main idea is to determine the best partitioning of a given CU between $2N \times 2N$ pixels and $N \times N$ pixels sub-blocks. Figure 2 illustrates the classification problem which predicts whether the CU of the depth d has to be merged in CU of the depth $d-1$.

It has been shown that the CTU partitioning during the RDO process is highly linked to the QP value and the texture complexity which can be statistically represented by the variance of blocks in Intra coding [1, 15, 40]. Figure 3 shows the CU boundaries of the 6th frame of *BasketballDrive* video sequence. It is worth noting that the regions with the lowest variance (smooth) tend to be encoded with larger blocks, as illustrated by the green circle in Fig. 3, while the blue circle shows a region with a high variance (high local activity), which are encoded with smaller blocks. In this section, we use this correlation between the pixel values of a block (variance) and its CTU partitioning to predict the quad-tree decomposition of a CTU and thus reduce drastically the encoding complexity.

3.1 Variance-Based Decision for Quad-Tree Partitioning

To study how to predict the quad-tree partitioning from the variance values of CU luminance samples, two populations of CUs at a current depth d are defined: *Merged (M)* and *Non Merged (NM)*. The CU belongs to the *Non Merged* population when the full RDO process chooses to encode the CU at the current depth d, while the CU belongs to the *Merged* population when the RDO process choose to encode the CUs at a new depth d' with $d' < d$. With a bottom-up approach (i.e. d from 4 to 1), all CUs of the quad-tree decomposition of all CTUs can be classified into one of these two populations.

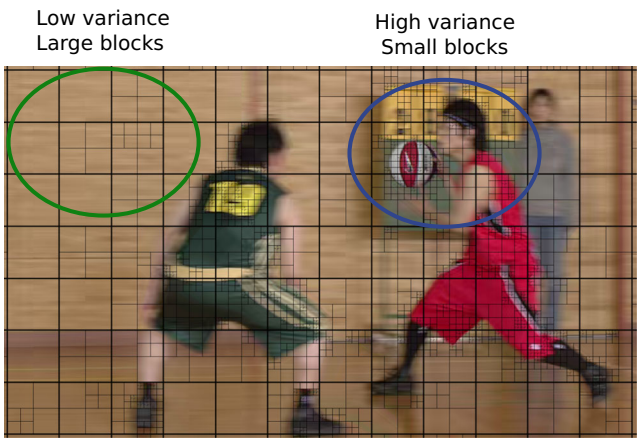


Figure 3 Quad-tree partitioning of the 6th HEVC intra coded frame of the *BasketballDrive* sequence. The green (resp. blue) circle shows that the lowest (resp. highest) variance regions tend to be encoded with larger (resp. smaller) units.

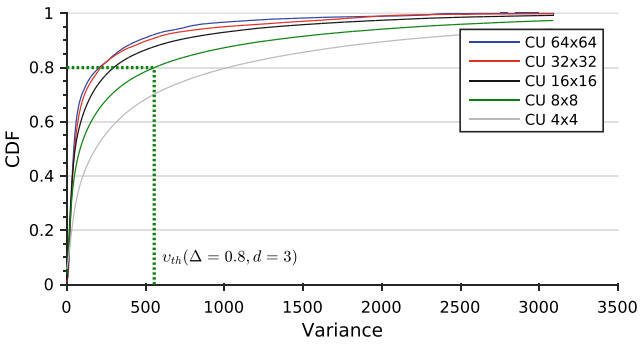


Figure 4 CDFs of the Merged population depending of CU size for the sixth frame of the sequence *BasketballDrive*. Under a specific probability Δ, a variance threshold can be extracted from the inverse CDF curve to classify a block as Merged.

Table 1 Variance thresholds $\upsilon_{th}(\Delta, d)$ of the 50th frame of two example sequences versus d and Δ.

Sequence name	Δ	$d=1$	$d=2$	$d=3$	$d=4$
PeopleOnStreet	0.3	31.8	31.1	51.4	97.0
	0.5	40.8	40.1	66.4	127.0
	0.7	49.8	50.1	84.4	166.0
	0.9	58.8	61.1	109.4	219.0
ParkScene	0.3	41.2	24.3	29.1	53.5
	0.5	51.2	31.3	37.1	70.5
	0.7	62.2	40.3	48.1	90.5
	0.9	74.2	50.3	62.1	117.5

Cumulative Distribution Function (CDF) of the *Non Merged* population can be used to decide whether a CU has to be merged or not. In our case, the CDF defines the probability of the variance population of a given CU size being less or equal to a given value.

Figure 4 shows the CDFs of CU variances depending on CU size for the sixth frame of the *BasketballDrive* video sequence. The CDF curves show that the probability for a CU size to be selected during the RDO process decreases when the variance of the CU increases. In other words, it is rare for a CU to have a variance greater than a certain threshold. From this observation, a variance threshold $\upsilon_{th}(\Delta, d)$ for each depth d can be extracted from the inverse CDF curve for a specific probability Δ. For example, Fig. 4 shows that 80% ($\Delta = 0.8$) of CUs 8×8 ($d = 3$) have a variance less than 555 represented by the green dotted lines in Fig. 4. Δ is the percentage of coding units whose variance is under the threshold υ_{th}, i.e. the variance threshold that triggers unit split.

Table 1 shows the thresholds $\upsilon_{th}(\Delta, d)$ for $d \in \{1, 2, 3, 4\}$ extracted from the CDFs for the 50th frame of the two sequence *PeopleOnStreet* and *ParkScene*. The Table illustrates that large variation of the threshold value across different video contents. In fact, the thresholds depend on the video contents and thus have to be determined on-the-fly from a Learning Frame (F_L).

3.2 Variance Threshold Modelling

Since the thresholds have to be adapted based on the video content, they have to be determined on-the-fly from Learning Frame, i.e. frames encoded with a full RDO process (unconstrained). The modelling of thresholds $\upsilon_{th}(\Delta, d)$ could have been conducted using variance PDFs with an approximation of the distribution based on a commonly known probability distribution but we observed that starting from a CDF curve is better performing [21]. An approximation of the thresholds directly from *Non Merged* population input features provides good results for the CDF curve. In Fig. 5, the variance thresholds $\upsilon_{th}(d = 3, \Delta = 0.8)$ is plotted versus the mean $\mu_\upsilon(d = 3)$ and the standard deviation $\sigma_V(d = 3)$ of CUs 8×8 variances. The values are extracted for 4 QP values 22, 27, 32 and 37 of 100 frames selected randomly from 5 different sequences: *BasketballDrive*, *BQTerrace*, *Cactus*, *ParkScene*, *PeopleOnStreet* and *Traffic*. Similar results are obtained for other CU sizes. The results show that for a fixed value of Δ, $\upsilon_{th}(\Delta, d)$ depends linearly on $\mu_\upsilon(d)$ and standard deviation $\sigma_\upsilon(d)$, and this independently from the QP value.

From this observation, $\upsilon_{th}(\Delta, d)$ can be modeled using the following linear relation:

$$\upsilon_{th}(\Delta, d) = a(\Delta, d)\cdot\mu_\upsilon(d) + b(\Delta, d)\cdot\sigma_\upsilon(d) + c(\Delta, d) \quad (2)$$

where $a(\Delta, d)$, $b(\Delta, d)$ and $c(\Delta, d)$ are coefficients modelling the threshold for each probability Δ and for each depth d. The coefficients are computed offline for each Δ and d values using a linear regression on all frames of *BasketballDrive*, *BQTerrace*, *Cactus*,*ParkScene*, *PeopleOnStreet* and *Traffic* for 4 values of QP 22, 27, 32 and 37. The *Rsq* is a metric that quantifies the accuracy of a predicting model. The average *Rsq* value is equal to 0.86, which confirms that the model fits the $\upsilon_{th}(\Delta, d)$, regardless of the video content and QP value.

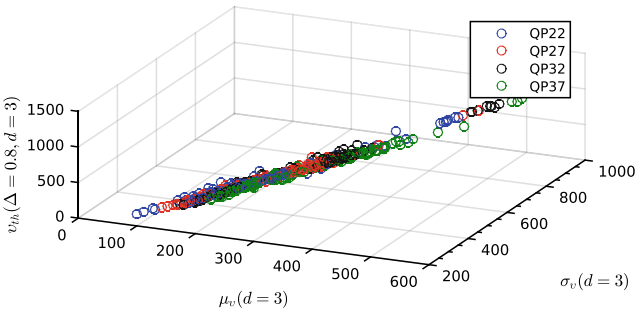


Figure 5 Variance thresholds $\upsilon_{th}(d = 3, \Delta = 0.8)$ versus the mean $\mu_\upsilon(d = 3)$ and the standard deviation $\sigma_V(d = 3)$ of CUs 8×8 variances. For a fixed value of Δ, the variance threshold values form a plane (independently of the QP value).

We can summarize the above analysis in the following steps:

- Thresholds from CDFs of variances can be predicted from reference Learning Frame (F_L).
- Look-Up Tables (LUTs) requires light computation and memory overheads for the determination of the threshold.
- The prediction of thresholds is independent from the QP value (Fig. 5).
- Threshold modelling is accurate with a mean Rsq of 0.86 for the different depths.
- Thresholds can be precomputed according to Δ value as a parameter.

The next section describes our first proposed algorithm to predict the CTU partitioning using a variance criterion and the obtained thresholds $\upsilon_{th}(\Delta, d)$.

3.3 Probabilistic Prediction of the CTU Partitioning

A description of the CTU partitioning is needed to explicitly depict the prediction of the quad-tree and then force the encoder to only encode this specific decomposition. Figure 6a illustrates the chosen representation of a CTU partitioning in the form of a CTU Depth Map (CDM) matrix 8 by 8. Each element of the matrix represents the depth d of a 8 by 8 square samples of the CTU. Since the CTU size is 64×64 and the minimum size of CU is 4×4, a matrix 8 by 8 can be used to describe all partitioning of a CTU. A depth of 4 in the CDM corresponds to 4 CU 4×4 in the CTU decomposition.

The following section describes the proposed algorithm and the prediction scheme that predicts in one-shot the CTU partitioning using a variance criterion and the thresholds $\upsilon_{th}(\Delta, d)$ described in Section 3.1.

3.3.1 Computing the CTU Depth Map

For a given CTU, let $\upsilon^d(i, j)$ be the variance of the luminance sample blocks of size $2^{6-d} \times 2^{6-d}$ at the depth level d and the local coordinates (i, j) into the CTU as illustrated in Fig. 7.

Algorithm 1 describes our proposed Probabilistic algorithm that predicts in one-shot the CTU partitioning. The algorithm takes as inputs the luminance samples of CTU and the table of thresholds υ_{th} previously computed by Eq. 2 to generate the CDM associated to the input CTU. In other words, the goal of this algorithm is to determine from the variance of the luminance samples the CDM matrix of the CTU. Then, the encoder only has to use the predicted depths instead of running an RDO process to encode the video, reducing significantly the complexity.

Algorithm 1 Probabilistic CTU Depth Map (CDM) generation

```
Data: Samples of CTU, υ_th(Δ, d)
Result: CDM matrix
1  CDM(x, y) = 4, ∀x, y ∈ [0, 7];
   // Initialization
2  Compute: υ^4(x, y), ∀x, y ∈ [0, 15];  // cf Eq. 3
3  for (d = 4; d > 0; d − −) do
4      δ = 2^(4−d);     // CDM matrix block size
5      for (y = 0; y < 8; y += δ) do
6          for (x = 0; x < 8; x += δ) do
               // Test if the 4 neighbor
                  blocks have the same
                  depth d when d < 4
7              if (CDM(x, y) = d &&
               CDM(x + ⌊δ/2⌋, y) = d &&
               CDM(x, y + ⌊δ/2⌋) = d &&
               CDM(x + ⌊δ/2⌋, y + ⌊δ/2⌋) = d) then
                   // Test if the variances
                      of blocks are lower
                      than the threshold
8                  x' = x/(2^(3−d)); y' = y/(2^(3−d));
                   if (υ^d(x', y') < υ_th(Δ, d) &&
                   υ^d(x' + 1, y') < υ_th(Δ, d) &&
                   υ^d(x', y' + 1) < υ_th(Δ, d) &&
                   υ^d(x' + 1, y' + 1) < υ_th(Δ, d))
                   then
                       // Block merging in the
                          CDM
9                      CDM(x + i, y + j) = d − 1,
                       ∀i, j ∈ [0, δ − 1];
10                     Compute: υ^d(x'/2, y'/2);
                       // cf Eq. 4
```

First of all, the full CDM is initialized with the depth value 4 (line 1) and all the variances $\upsilon^4(x, y)$ of the CU 4×4 (line 2) are computed using Eq. 3 where $p_{x,y}(i, j)$ is the luminance component of the samples at the coordinate (i, j) in the CU 4×4 at the position (x, y) and $\bar{p}_{x,y}$ the average value of the block.

$$\upsilon^4(x, y) = \frac{1}{16} \sum_{i=0}^{3} \sum_{j=0}^{3} (p_{x,y}(i, j) - \bar{p}_{x,y})^2 \tag{3}$$

Then, the algorithm explores the CTU decomposition with a bottom-up approach: from $d = 4$ to $d = 1$ (line 3). For the current depth d, the algorithm browses the CDM (lines 5–6) taking the block size δ in the CDM (line 4) into account. Afterwards, the algorithm tests if the 4 neighbor

blocks in the Z-scan order have the same depth d, except when $d = 4$ (line 7). The algorithm does not try to merge neighbor blocks if they have different depths as illustrated in Fig. 8.

Algorithm 2 CTU depth map refinement

```
   Data: CDM matrix
   Result: RCDM matrix
1  if CDM(0, 0) = 0 then
2      RCDM(x, y) = 0, ∀x, y ∈ [0, 7]; // Set RCDM at 0
3  else
4      for (y = 0; y < 8; y ++) do
5          for (x = 0; x < 8; x ++) do
6              if CDM(x, y) = 4 then
7                  RCDM(x, y) = 3    // Automatic merge
8              else
9                  d = CDM(x, y)        // Get the depth
10                 δ = 2^(4−d)  // CDM matrix block size
                   // Test if it is the last blocks in the Z-scane order for the depth d
11                 if ((x % δ) = δ/2) && ((y % δ) = δ/2) then
                       // Test if the three other blocks have the same depth
12                     if (CDM(x − δ/2, y) = d &&
                       CDM(x, y − δ/2) = d &&
                       CDM(x − δ/2, y − δ/2) = d) then
13                         RCDM(x − δ/2 + i, y − δ/2 + j) = d, ∀i, j ∈ [0, δ − 1];// Fill the RCDM
```

Since the algorithm is bottom-up, there is no need to test the condition when $d = 4$ because the CDM is initialized at $d = 4$ which is the starting depth of the algorithm. If the previous condition is true, then the algorithm tests whether the blocks have to be merged or not using the variance criteria (line 8) previously detailed in Section 3.1. If the 4 blocks variances υ^d are lower than the given threshold $\upsilon_{th}(\Delta, d)$ then the blocks are merged and the corresponding elements in the CDM are set to $d - 1$ (line 9) and the variance of the merged block is calculated three times using the combined variance Eq. 4.

$$\upsilon_{a\cup b} = \frac{(2n-1)(\upsilon_a + \upsilon_b) + n(\mu_a - \mu_b)^2}{4n-1} \tag{4}$$

Equation 4 computes the variance of two sets of data a and b containing the same number of observations n with μ and υ corresponding to the mean and the variance of the specified data set, respectively [5].

3.3.2 Refining the CTU Depth Map

To increase the accuracy of the one-shot depth map prediction with a limited impact on the complexity, a second algorithm is designed that refines the CDM.

The algorithm, described by Algorithm (2), takes as input a CDM matrix from Algorithm (1) and generates a second CDM called RCDM. The RCDM is the result of merging all groups of four neighboring blocks (in the Z-scan order) having the same depth in the input CDM. Algorithm (2) details the process as follows.

The first step checks whether the input CDM depth is equal to 0, if so then no merge can be applied and thus the RCDM is also set to 0 (line 2). If not, the CDM is analysed element by element (lines 4–5). Due to the fact that a depth of 4 in a CDM corresponds to 4 CUs 4×4, they are always merged to a depth 3 and thus the value in the RCDM is automatically set to 3 (line 7). For the general case (i.e $d \in \{1, 2, 3\}$), if the evaluated element in the matrix correspond to the fourth block (in the Z-scan order) of the given depth d (line 11) and if the 3 others blocks have the depth d (line 12), then the algorithm fills the corresponding blocks of the RCDM with the upper depth $d - 1$ (line 13).

Figures 6 show an example of a CDM (Fig. 6a) and its associated RCDM (Fig. 6b) matrices. The grey blocks in the RCDM Fig. 6b represent the merged blocks. The next section describes our Probabilistic energy reduction scheme.

3.3.3 Resulting Probabilistic CTU Prediction Method

Based on the previous elements, we propose to limit the recursive search of the RDO process on the CTU quad-tree decomposition by predicting the coding-tree partitioning from video frame content properties. We introduce a probabilistic variance-aware quad-tree partitioning prediction method, illustrated in Fig. 9. First, the video sequence is split into Groups of Frames (GOF). The first frame of a GOF, called Learning Frame (F_L) is encoded with a full RDO process. From this encoding are extracted the variances υ^d according to the depth $d \in \{1, 2, 3, 4\}$ selected during the full RDO process. Then, the two following statistical moments according the depth d are computed: the means μ_{υ^d} and the standard deviations σ_{υ^d} of the variance populations υ^d. According to the parameter Δ, the set of thresholds $\upsilon_{th}(d)$ are calculated using Eq. 2 and the LUT of the coefficients $a(\Delta, d)$, $b(\Delta, d)$ and $c(\Delta, d)$ computed off-line (cf. Section 3.2).

Figure 6 CDM and its associate RCDM matrices of the CTU partitioning of Fig. 1.

3	3	2	2	1	1	1	1
3	3	2	2	1	1	1	1
2	2	3	3	1	1	1	1
2	2	3	4	1	1	1	1
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2

(a) CTU Depth Map (CDM)

2	2	2	2	1	1	1	1
2	2	2	2	1	1	1	1
2	2	3	3	1	1	1	1
2	2	3	3	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

(b) Refined CTU Depth Map

The other frames of the GOF, called *constrained frames* (F_C), are encoded with a limited RDO process. For each CTU, Algorithm (1) is applied using the sets of thresholds previously computed for the F_L to generate the CDM. However, the CDM generated with Algorithm (1) is very restrictive for the RDO process as it allows only one depth to be searched for each CU of a CTU. To increase the accuracy of the depth map prediction with limited impact on the complexity, the CDM is then refined by Algorithm (2) to generate a second CDM called RCDM. To finish, the HEVC encoder is forced to only apply the RDO process between the two CDMS.

To conclude this section, our proposed probabilistic energy reduction scheme takes as input the parameter Δ to generate CDMS and RCDM. Then, the HEVC encoder is forced to only apply the RDO process between the CDM and the RCDM. The next section details the competing Machine Learning method.

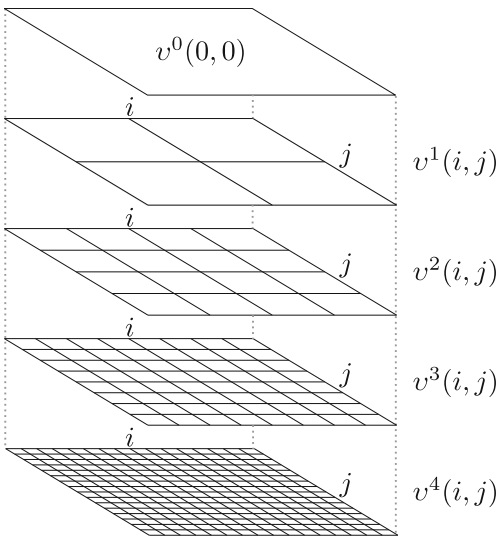


Figure 7 $v^d(i, j)$: variance of the luminance sample blocks of size $2^{6-d} \times 2^{6-d}$ of a CTU versus the depth level d.

4 Machine Learning Approach for Predicting an HEVC Quad-Tree Partitioning

This Section presents our second quad-tree prediction method based on Machine Learning. This quad-tree prediction is then used to drastically simplify the brute force algorithm usually employed in HEVC encoders.

4.1 Machine Learning Based Decision

As in the probabilistic method (Section 4), the Machine Learning-based quad-tree prediction follows a bottom-up approach (from CU 4×4 to CU 32×32). The classification problem remains to determine whether the CU of the depth d has to be merged in CU of the depth $d - 1$ as illustrated in Fig. 2. The next section details the training set-up of the learning algorithm.

4.1.1 Training Set-Up for the Coding Tree Structure Determination

Machine Learning efficiency is very linked to the diversity of data serving for the training. Video sequences used to train the Machine Learning framework are chosen to cover a vast space of content types. To select this training data set including a large range of video contents and complexities, the Spatial Information (SI) and Temporal Information (TI) metrics [12] are used to characterize video sequences. The TI and SI give respectively the degrees of motion and spatial details in the video sequence. Since compression complexity is highly linked to these two spatio-temporal parameters, the set of training sequences for the Machine Learning feature evaluation should span a large range of both SI and TI.

Figure 10 shows the SI and TI for the video sequences according to the classes (from A to E). The chosen training sequence set (circled in Fig. 10) is composed of one video sequence of each class, well distributed in term of SI and

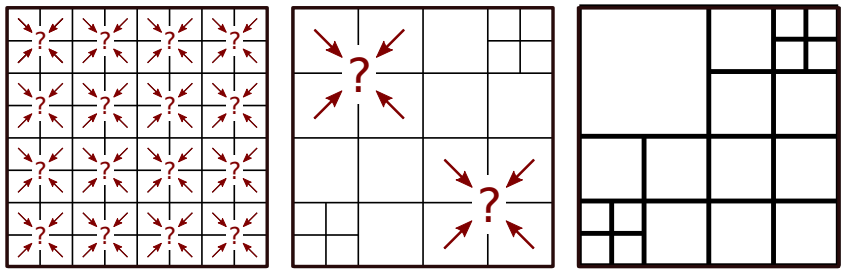


Figure 8 Probabilistic bottom-up Algorithm (1) example. Algorithm (1) does not try to merge neighbor blocks if they have different depths.

TI: *Traffic* (class A), *Cactus* (class B), *BQMall* (class C), *BasketballPass* (class D) and *Johnny* (class E).

Overfitting, i.e. overspecializing a model to a training set, constitutes one of the main risks for the quality of an Machine Learning-based model [6]. Thus, the dataset used for training should result in a low bias. In our case, due to the broad range of resolutions and frame rates across the training sequences, the total number of CTU for each class is not equally distributed. For instance, sequences with high resolution contain a high number of CTUs with low texture complexities when compared to sequences with low resolution. To avoid such bias, datasets used for training are forced to be composed of a fixed number of CTUs from each class. To avoid the temporal bias, which would lead to redundant information, the sampled CTUs come from frames uniformly distributed throughout the sequences: 13 frames of the class A, 25 frames of class B, 55 frames for class E, 125 frames of class C and 500 frames of class D. For each depth d, 80000 instances are randomly sampled from the previous defined data pool, composed by 40000 instances of each prediction decision at each depth d.

The open source Waikato Environment for Knowledge Analysis (WEKA) Machine Learning framework is used for the training process [11]. Weka is chosen for its popularity and extensive set of documentations. It includes a large number of Machine Learning algorithms for data mining tasks, such as *REPTree*, *LMT*, *RandomForest*, *BFTree* and *C4.5* among others. WEKA also provides several useful tools for features evaluation that use strategies according to a search algorithm so as to rank the features depending on their usefulness. For the current study, features have been selected using the *information gain* provided by the WEKA software. *Information gain* is based on the Kullback-Leibler Divergence (KLD) [18], also called relative entropy, which measures the divergence between two probability distributions.

4.1.2 Decision Trees-Based Partitioning Decisions

State-of-the-art studies described in Section 2.3 gather many characteristics used to predict the coding tree decomposition of a CTU. To predict the coding tree in one-shot, only characteristics independent from the encoding process with a limited overhead of computation are considered.

The choice of these features is detailed in [23]. They have been deduced from an extensive study of two factors: the *information gain* provided by the WEKA software and the overhead of computation under a real-time encoder. The features vector for CU at coordinates (x, y) and depth d of a given CTU, $\mathcal{F}^d_{x,y}$, is composed of the following 12 features:

- **CU var** [7, 15, 21, 25, 29, 30] : the variance of the CU luminance samples of depth d (1 features).
- **Lower-CU var** [7, 15, 21, 29, 30]: the variances of the 4 sub-CU luminance samples of depth $d + 1$ (4 features).

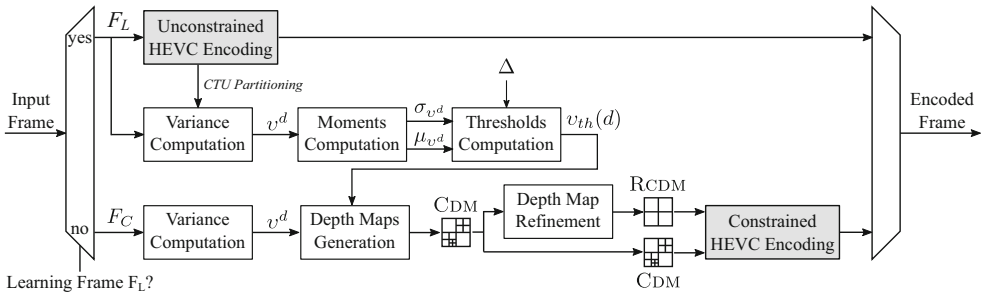


Figure 9 Diagram of the Probabilistic proposed energy reduction scheme. The Learning Frames (F_L) are encoded with a full RDO process (unconstrained) and the block variances of the resulting quad-tree decomposition are used to compute the set of thresholds $\upsilon_{th}(d)$. The constrained frames (F_C) use these thresholds to generate the CDM. The CDM is then refined to generate the Refined CTU Depth Map RCDM and the encoder is finally constrained to only apply the RDO process in the interval formed by the two CDMs.

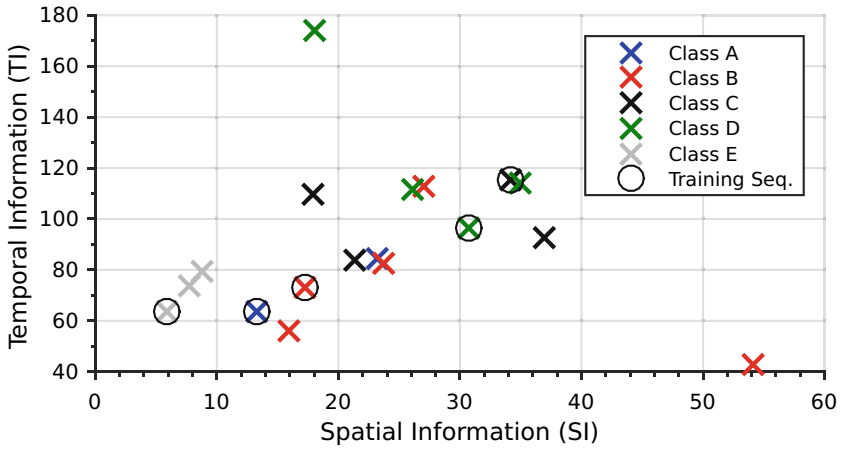


Figure 10 SI and TI of the video sequences according the classes. The chosen training sequence set is circled.

- **Upper-CU var** [15, 21, 25, 29, 30]: the variances of the upper CU luminance samples of depth $d-1$ (1 features).
- **Nhbr-CU var** [7, 15]: the variances of the neighbouring CU luminance samples of the depth d in the Z-scan order (3 features).
- **Var of lower-CU mean** [29, 30]: the variance of the mean of the 4 sub-CU luminance samples of the depth $d+1$ (1 feature).
- **Var of lower-CU var** [29, 30]: the variance of the variance of the 4 sub-CU luminance samples of the depth $d+1$ (1 feature).
- **QP**: the QP of the frame (1 feature).

The training of the decision trees is performed with the C4.5 algorithm [28] because the trees it generates are light weight. In terms of information gain, the C4.5 algorithm uses KLD to select the best features for each decision. The C4.5 algorithm is iterating among all training instances and searches for each features the threshold that achieves the best classification, i.e. with the highest information gain. Then, the features and its corresponding threshold are used to divide the training instances into two subsets. Finally, the process is recursively iterated on the two different subsets of training instances.

To measure the accuracy of the decision trees, a 10-fold cross-validation is performed on the training instances. The cross-validation technique evaluates a predictive models by partitioning the original instances into a training set to train the model, and a test set to evaluate it. In 10-fold cross-validation, the original instances are randomly split into 10 equally sized subsets. Among the 10 subsets of instances, one subset is used as the validation instances for testing the model, and the remaining 9 subsets are used as training instances. The cross-validation process is then repeated 10 times (called folds), with each of the 10 subsets used exactly once as the validation instances. Let the Percentage of Correctly Classify Instances (PCCI) given by the 10-fold cross-validation be the accuracy of the decision trees.

Two types of classifiers are defined for each depth d: the *Merge* and *Split* decision trees. These two decision trees solve the same classification problem illustrated in Section 3 by Fig. 2 but differ in their input features. The Merge decision trees use the features linked to CU of depth d to predict if the 4 CUs of the d have to be merged in the CU of depth $d-1$. The Split decision trees use the features linked to CU of depth $d-1$ to predict if the current CU of the $d-1$ have to be split in 4 CUs of depth d.

Table 2 Decision trees dimensions and accuracy (PCCI) according to the depth d. The accuracy of both Merge and Split decision trees are over than 80% of good decisions

	Merge decision trees			
Depth	$d=4$	$d=3$	$d=2$	$d=1$
Nb leaves	18	15	10	9
Size	15	13	13	15
PCCI	81.39%	80.52%	80.19%	81.26%
	Split decision trees			
Depth	$d=3$	$d=2$	$d=1$	$d=0$
Nb leaves	18	15	10	9
Size	35	29	19	17
PCCI	82.24%	80.89%	80.87%	80.83%

Table 2 summarizes the trained tree sizes, number of leaves and the PCCI of the 8 decision trees. Results in Table 2 show that the accuracy of both Merge and Split decision trees are over than 80% of good decisions.

The next sections describes how we use decisions trees to predict CTU partitioning.

4.2 Formalisation of the CTU Partitioning Decisions

Let $P_S(\mathcal{F}^d_{x,y})$ and $P_M(\mathcal{F}^d_{x,y})$ respectively be the prediction results of the *Split* and *Merge* trees for the features vector $\mathcal{F}^d_{x,y}$. The prediction decision $\mathbb{D}_d(x, y)$ is defined as the prediction resulting of the best combination of the decision trees at the depth d by Eq. 5:

$$\mathbb{D}_d(x,y) = \begin{cases} \mathbb{P}_M^{\{1,2\}}(x,y) \wedge (\overline{P_S}(\mathcal{F}^{d-1}_{\lfloor \frac{x}{2} \rfloor, \lfloor \frac{y}{2} \rfloor}) & \text{if } d \in \{1,2\} \\ \mathbb{P}_M^{\{3,4\}}(x,y) \vee (\overline{P_S}(\mathcal{F}^{d-1}_{\lfloor \frac{x}{2} \rfloor, \lfloor \frac{y}{2} \rfloor}) & \text{if } d \in \{3,4\} \end{cases} \tag{5}$$

with $\mathbb{P}_M^{\{1,2\}}(x, y)$ and $\mathbb{P}_M^{\{3,4\}}(x, y)$ such as:

$$\begin{aligned} \mathbb{P}_M^{\{1,2\}}(x,y) \;=\; & P_M(\mathcal{F}^d_{x,y}) \wedge P_M(\mathcal{F}^d_{x+1,y}) \\ & \wedge P_M(\mathcal{F}^d_{x,y+1}) \wedge P_M(\mathcal{F}^d_{x+1,y+1}) \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbb{P}_M^{\{3,4\}}(x,y) \;=\; & P_M(\mathcal{F}^d_{x,y}) \vee P_M(\mathcal{F}^d_{x+1,y}) \\ & \vee P_M(\mathcal{F}^d_{x,y+1}) \vee P_M(\mathcal{F}^d_{x+1,y+1}). \end{aligned} \tag{7}$$

In other words, for the high depth $d \in \{1, 2\}$, the algorithm will merge the four CUs at the depth d if all the five decision trees predict to merge. In contrast, for the low depth $d \in \{3, 4\}$, the algorithm will merge the four CUs at the depth d if at least one decision tree predicts to merge.

4.2.1 Machine Learning Prediction Algorithm for CTU Partitioning

Algorithm (3) describes our proposed bottom-up algorithm that predicts the CTU partitioning using a Machine Learning approach. The algorithm takes as inputs all the features $\mathcal{F}^d_{x,y}$ defined in Section 4.1.2 previously computed to generate the CDM associated to the input CTU.

First of all, the full CDM is initialized with depth value 4 (line 1). Then, the algorithm explores the CTU decomposition with a bottom-up approach: from $d = 4$ to $d = 1$ (line 2). For the current depth d, the algorithm browses the CDM (lines 4–5) taking the block size δ in the CDM (line 3) into account. Afterwards, the Machine Learning Algorithm (3) differs from the probabilistic Algorithm (1) and does not test if the 4 neighbor blocks in the Z-scan order have the same depth d, as illustrated in Fig. 11. Indeed, better results are obtained without this condition, contrary to the probabilistic case.

Then the algorithm tests whether the blocks have to be merged or not using the merge and split decision trees prediction respectively $P_M(\mathcal{F}^d_{x,y})$ and $P_S(\mathcal{F}^d_{x,y})$ previously detailed in Section 4.1.2 and the combination defined by Equation 5 (line 7). If the prediction is true, the blocks are merged and the corresponding elements in the CDM are set to $d - 1$ (line 8).

Algorithm 3 Machine learning CTU depth map inference

Input: $\mathcal{F}^d_{x,y}$ `// All features of the CTU`
Result: CDM matrix

1 $\text{CDM}(x, y) = 4, \forall x, y \in [0, 7]$;
`// Initialization`
2 **for** $(d = 4; d > 0; d --)$ **do**
3 $\quad \delta = 2^{4-d}$; `// CDM matrix block size`
4 $\quad$ **for** $(y = 0; y < 8; y \mathrel{+}= \delta)$ **do**
5 $\quad\quad$ **for** $(x = 0; x < 8; x \mathrel{+}= \delta)$ **do**
6 $\quad\quad\quad x' = x/\delta; y' = y/\delta;$ `// F idx`
`// Test the condition of merge decision cf eq. 5, 6 and 7`
7 $\quad\quad\quad$ **if** $\mathbb{D}_d(x', y')$ **then**
`// Block merging in the CDM`
8 $\quad\quad\quad\quad \text{CDM}(x + i, y + j) = d - 1,$
$\quad\quad\quad\quad \forall i, j \in [0, \delta - 1]$;

4.2.2 Resulting Machine Learning CTU Prediction Method

Figure 12 presents a high-level diagram of our resulting Machine Learning CTU partitioning prediction technique. Thanks to the offline training of the decision trees, all the frames are constrained and no learning frame is needed, in contrast to the probabilistic approach (cf Section 3). The features detailed in Section 4.1.2 are computed for the whole frame to minimize the computational complexity overhead. Then, Algorithm (3) is applied using the features to generate the CDM. As with the Probabilistic approach, to increase the accuracy of the depth map prediction with limited impact on the complexity, the CDM is refined using Algorithm 2 to generate the RCDM. Finally, the HEVC encoder is forced to only apply the RDO process between the previously generated CDM and RCDM.

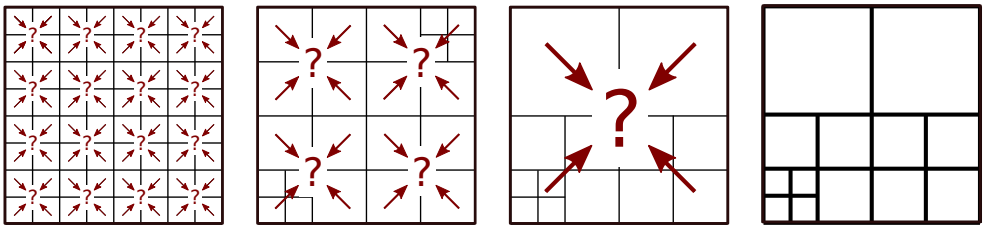


Figure 11 Machine Learning bottom-up Algorithm (3) example. Algorithm (3) do not take into account the decisions taken at the previous depth.

5 Probabilistic Approach versus Machine Learning for One-Shot Quad-Tree Prediction

This section gives the experimental setup and the results obtained for the two proposed energy reduction schemes on the real time HEVC encoder *Kvazaar* [36].

5.1 Experimental Set-Up and Metrics to Evaluate the Quad-Tree Partitioning Predictions

5.1.1 Experimental Set-Up and Parameters

To conduct the experiments, 18 video sequences [2] that strongly differ from one another in terms of frame rate, motion, texture and spatial resolution were used. All experimentations are performed on one core of the *EmETXe-i87M0* platform from *Arbor Technologies* based on an Intel Core i5-4402E processor at 1.6 GHz. The used HEVC software encoder is the real time Kvazaar [16, 17, 39] in All Intra (AI) configuration. Since the configuration aims to be real-time, from [20], the Rate-Distortion Optimisation Quantization (RDOQ) [14] and the Intra transform skipping [19] features are disabled. Each sequence is encoded with 4 different QP values: 22, 27, 32, 37 [2]. For the Probabilistic approach, previous experiments showed that the best prediction is obtained with $\Delta \in [0.6, 0.7]$ [21]. For the following experiments, Δ is fixed to 0.6 and GOF size is fixed to 50, which is shown in [22] to be an appropriate value for drastic energy reductions.

Bjøntegaard Delta Bit Rate (BD-BR) and Bjøntegaard Delta Psnr (BD-PSNR) [41] are used to measure the compression efficiency difference between two encoding configurations. The BD-BR reports the average bit rate difference in percent for two encodings at the same quality in terms of Peak Signal-to-Noise Ratio (PSNR). Similarly, the BD-PSNR measures the average PSNR difference in decibels (dB) for two different encoding algorithms considering the same bit rate.

To measure the energy consumed by the platform, Intel Running Average Power Limit (RAPL) interfaces are used to obtain the energy consumption of the CPU package, which includes cores, IOs, DRAM and integrated graphic chipset. As shown in [10], RAPL power measurements are coherent with external measurements and [8] proves the reliability of this internal measure across various applications. In this work, the power gap between the IDLE state and video encoding is measured. The CPU is considered to be in IDLE state when it spends more than 90% of its time in the C7 C-states mode. The C7 state is the deepest C-state of the CPU characterized by all core caches being flushed, the PLL and core clock being turned off as well as all uncore domains. The power of the board is measured to 16.7W when the CPU is in idle mode and goes up to 31W during video encoding in average. RAPL shows that 72% of this gap is due to the CPU package, the rest of the power going to the external memory, the voltage regulators and other elements of the board.

5.1.2 Experimental Metrics

The performance of the proposed energy reduction scheme is evaluated by measuring the trade-off between Energy Reduction (ER) in % and Rate-Distortion (RD) efficiency using the BD-BR and BD-PSNR. ER is defined by Eq. 8:

$$ER = \frac{100}{4} \sum_{QP_i \in \{22,27,32,37\}} \frac{E_{Ref}(QP_i) - E_{red}(QP_i)}{E_{Ref}(QP_i)} \quad (8)$$

where $E_{Ref}(QP_i)$ is the energy spent to encode the video sequence without constraint and $E_{red}(QP_i)$ the energy

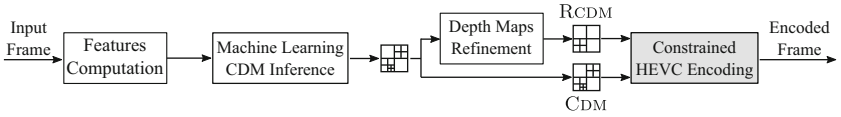


Figure 12 Diagram of the Machine Learning proposed energy reduction scheme. The features are firstly computed for the whole input frame. The features are using to generate the CDM. The CDM is then refined to generate the RCDM and the encoder is finally constrained to only apply the RDO process in the interval formed by the two CDMS.

to encode the same sequence with our proposed energy reduction scheme, both with $QP = QP_i$.

The main objective of Algorithms (1) and (3) is to generate a CDM that minimises the prediction error when compared to what the full RDO process would generate. To evaluate the accuracy of our predictions, we define the normalized L_1 *distance* Γ between two CDMS in terms of depth levels as follow:

$$\Gamma(A, B) = \left[\sum_{i=0}^{7} \sum_{j=0}^{7} |A(i, j) - B(i, j)| \right] /64 \quad (9)$$

where A and B are the two compared CDMS. In other words, the metric $\Gamma(A, B)$ measures the average gap in number of depth levels between two CDMS A and B of a given CTU. Let use Fig. 6 as example, the distance Γ between the CDM Fig. 6a, b is equal to $\Gamma = (4 + 1 + 16)/64 = 0.3281$.

In addition to the *distance* metric, we define the *recall* ρ between two CDMS A and B:

$$\rho(A, B) = \left[\sum_{i=0}^{7} \sum_{j=0}^{7} \delta(A(i, j) - B(i, j)) \right] \times \frac{100}{64}, \quad (10)$$

$$\text{with } \delta(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases} \quad (11)$$

The recall $\rho(A, B)$ represents the share of correct quad-tree decomposition in term of pixel area between predicted CTUs A and reference CTUs B. Let us use Fig. 6 as example, the recall between the CDM Fig. 6a (considered as predicted) and Fig. 6b (considered as reference) is equal to $\rho = 43 \times \frac{100}{64} = 67.19\%$.

The recall $\rho(P, R)$ and the distance $\Gamma(P, R)$ are used in the following sections to evaluate the accuracy of the prediction with P being the predicted CDM and R the reference CDM,[1] generated by a full RDO process (optimal). The average of $\rho(P, R)$ measurements gives the percentage of good prediction in term of pixel area, it falls between 0% and 100% and the more $\rho(P, R)$ is close to 100%, the more the predicted CDMS accurately fit the reference CDMS. The average distance $\Gamma(P, R)$ represents the mean error in term of depth between the predicted CDMS and the reference one, the more $\Gamma(P, R)$ is close to 0, the more precise the predicted CDM P becomes.

5.2 Comparison of Probabilistic and Machine Learning Approach for Predicting an HEVC Quat-Tree Partitioning

Table 3 details the performance of the two proposed energy reduction schemes: the Probabilistic and the Machine Learning approaches, for 18 different sequences belonging to the 5 classes A, B, C, D and E, each one corresponding to a specific resolution or video content. Two types of metrics are detailed in Table 3. The first one is composed by the $\rho(P, R)$ measure and the distance $\Gamma(P, R)$ defined in Section 5.1.2 averaged across the four QP values which evaluate the precision of the quad-tree prediction for all constrained frames. The second one is composed by BD-BR and BD-PSNR [41] that are the common metrics used in video compression to measure the compression efficiency difference between two encodings. The ER values include the energy overhead due to the entire energy reduction scheme (features or variance computation and CDM prediction). In a real-time configuration (see Section 5.1.1), the computational overheads in the real-time encoder Kvazaar of our two proposed methods are between 1% and 1.9% for the Probabilistic approach and between 1.5% and 2.5% for the Machine Learning approach.

In terms of quad-tree prediction accuracy in one-shot, Table 3 shows that the Machine Learning energy reduction techniques achieve better results (around 53% of $\rho(P, R)$ for a distance $\Gamma(P, R)$ of 0.67 depth level) than the Probabilistic energy reduction techniques (around 50% of $\rho(P, R)$ for a distance $\Gamma(P, R)$ of 0.79 depth level).

The results show that both energy reduction techniques achieve an average of 58% of energy reduction. In fact, the overhead due to the unconstrained Learning Frame (F_L) and the variance computations of the Probabilistic approach is approximately equal to the overhead of the features computations of the Machine Learning approach. However, even if the Probabilistic approach does not constrain all the frames (only 49 every 50 frames), this approach causes more encoding degradations: +0.33% of BD-BR and -0.02dB of BD-PSNR, than the Machine Learning approach. These results show that the two metrics $\rho(P, R)$ and $\Gamma(P, R)$ of quad-tree prediction accuracy are well correlated with the impact in encoding degradations.

It is noticeable in Table 3 than the Kimono sequence has more degradations than the other sequences: 13.28% of BD-BR increasing with the Probabilistic approach and 9.51% of BD-BR increasing with the Machine Learning approach. This can be explained by the texture specificity of the Kimono video sequence which is composed by a traveling of trees and vegetation in the background. This video sequence has the highest Spatial Information (54.1) due to the details. Nevertheless, the results show that the Machine Learning approach reduces the degradation of 3.77% of BD-BR compare to the Probabilistic approach.

The performance of state-of-the-art solutions (cf Section 2.3) based on HM can not be directly compared to these results. Indeed, they are measured comparatively to a large compression time, far from real-time. The complexity overhead of state-of-the-art solutions is thus comparatively higher in the context of a real-time encoder. Previously

[1] Exhaustive search leading to the optimal solution.

Table 3 The recall $\rho(P, R)$, distance $\Gamma(P, R)$, BD-BR, BD-PSNR and ER of the Probabilistic and Machine Learning drastic energy reduction schemes according to the sequences. For the same energy reduction, the Machine Learning energy reduction techniques achieve better results than the Probabilistic energy reduction techniques for both quad-tree prediction accuracy and encoding degradation.

	Probabilistic					Machine learning				
Sequence	ρ (in %)	Γ (in d)	BD-BR (in %)	BD-PSNR (in dB)	ER (in %)	ρ (in %)	Γ (in d)	BD-BR (in %)	BD-PSNR (in dB)	ER (in %)
Traffic	44.76	0.86	4.59	-0.24	60.16	46.96	0.79	4.05	-0.21	58.69
PeopleOnStreet	51.92	0.70	4.28	-0.24	59.06	51.34	0.72	3.73	-0.21	57.09
Kimono	18.87	2.06	13.28	-0.43	52.59	40.82	0.94	9.51	-0.31	61.33
ParkScene	38.66	1.24	4.29	-0.19	55.84	47.09	0.84	3.86	-0.17	60.28
BasketballDrive	47.68	0.76	3.69	-0.11	60.75	50.74	0.69	4.65	-0.13	60.16
Cactus	43.86	0.95	3.56	-0.13	60.40	50.03	0.73	3.79	-0.14	61.34
BQTerrace	51.83	0.66	2.25	-0.14	62.06	50.35	0.69	1.99	-0.13	58.93
RaceHorses480	46.86	0.91	3.11	-0.18	59.68	54.59	0.64	2.95	-0.17	60.37
PartyScene	55.38	0.56	1.88	-0.13	59.48	52.54	0.63	1.10	-0.08	57.10
BasketballDrill	51.40	0.64	2.74	-0.13	58.33	43.62	0.85	4.97	-0.24	62.26
BQMall	53.78	0.66	3.46	-0.19	57.54	52.62	0.66	3.16	-0.17	57.13
RaceHorses240	52.01	0.69	2.47	-0.15	59.07	57.88	0.54	1.93	-0.12	58.57
BQSquare	65.92	0.41	2.73	-0.21	57.81	67.81	0.42	1.00	-0.08	55.51
BlowingBubbles	55.70	0.55	1.45	-0.10	55.25	51.90	0.64	1.24	-0.08	53.54
BasketballPass	57.55	0.55	2.39	-0.14	59.20	57.46	0.56	2.31	-0.14	58.01
FourPeople	49.80	0.74	4.78	-0.27	55.70	50.03	0.73	4.51	-0.25	55.13
Johnny	53.61	0.67	5.76	-0.23	51.63	60.17	0.55	5.49	-0.22	52.26
KristenAndSara	58.50	0.56	4.10	-0.21	54.24	61.32	0.50	4.62	-0.23	53.39
Average	**49.89**	**0.79**	**3.93**	**-0.19**	**57.71**	**52.63**	**0.67**	**3.60**	**-0.17**	**57.84**

published results can thus not be directly applied to reduce the energy consumption in a real-time encoder as the two methods developed here.

To conclude, the Machine Learning approach achieves better results on average than the probabilistic approach and does not require unconstrained learning frames to predict the quad-tree partitioning. These two points make the proposed Machine Learning approach a good candidate to build energy reduction methods for real-time HEVC encoders.

6 Conclusion

This paper proposes and compares two energy reduction methods for real-time HEVC Intra encoders. These methods are based on CTU partitioning prediction techniques that drastically limit the recursive RDO process. The first proposed method exploits the correlation between a CTU partitioning and the variance of the CTU luminance samples to predict the quad-tree decomposition in one-shot. The second method uses a Machine Learning method to predict in one-shot the quad-tree decomposition.

Experimental results show that the Machine Learning method has a slight edge over the probabilistic method and that this performance has a direct impact on the encoding degradations. Both energy reduction techniques are capable of reducing the energy consumption of the HEVC encoder by 58% — including the additional algorithm overhead under a real-time encoder — for a bit rate increase of respectively 3.93% and 3.6%. The obtained energy gain is substantial and close to the theoretical maximum of 78% gain that would be obtained if the perfect quad-tree decomposition would be known in advance. Future work will use one-shot quad-tree partitioning prediction to control the energy consumption of an HEVC Intra encoder for a given energy consumption budget.

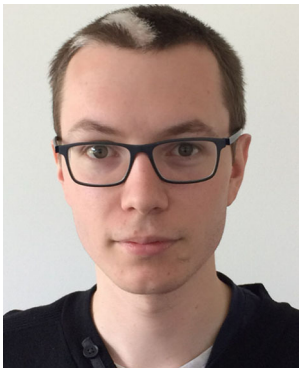
Acknowledgments This work is partially supported by the French ANR ARTEFaCT project, by COVIBE project funded by Brittany region and by the European Celtic-Plus project 4KREPROSYS funded by Finland, Flanders, France, and Switzerland.

References

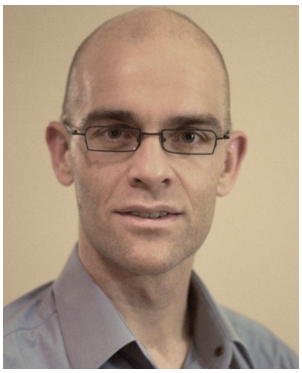
1. Biao, M., & Cheung, R.C.C. (2015). A fast CU size decision algorithm for the HEVC intra encoder. *IEEE Transactions on Circuits and Systems for Video Technology*, *25*(5), 892–896. https://doi.org/10.1109/TCSVT.2014.2363739.
2. Bossen, F. (2013). Common HM test conditions and software reference configurations. In *JCTVC-L1100*. Switzerland: Geneva.
3. Carroll, A., & Heiser, G. (2010). An analysis of power consumption in a smartphone. In *USENIX annual technical conference, Boston, MA* (Vol. 14, pp. 21?21).
4. Cassa, M.B., Naccari, M., Pereira, F. (2012). Fast rate distortion optimization for the emerging HEVC standard. In *Picture coding symposium (PCS), 2012* (pp. 493–496). IEEE
5. Chan, T.F., Golub, G.H., LeVeque, R.J. (1982). Updating formulae and a pairwise algorithm for variances computing sample. In *COMPSTAT 1982 5th symposium held at Toulouse 1982* (p. 30). Springer Science & Business Media.
6. Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87. https://doi.org/10.1145/2347736.2347755.
7. Duanmu, F., Ma, Z., Wang, Y. (2015). Fast CU partition decision using machine learning for screen content compression. In *2015 IEEE international conference on image processing (ICIP)* (pp. 4972–4976). IEEE.
8. Efraim, R., Alon, N., Doron, R., Avinash, A., Eliezer, W. (2012). Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Computer Society*, *32*(2), 20–27.
9. Feng, L., Dai, M., Zhao, C.l., Xiong, J.y. (2016). Fast prediction unit selection method for HEVC intra prediction based on salient regions. *Optoelectronics Letters*, *12*(4), 316–320. https://doi.org/10.1007/s11801-016-6064-8.
10. Hackenberg, D., Schone, R., Ilsche, T., Molka, D., Schuchart, J., Geyer, R. (2015). An energy efficiency feature survey of the intel Haswell processor. In *2015 IEEE international parallel and distributed processing symposium workshop (IPDPSW)* (pp. 896–904). IEEE. https://doi.org/10.1109/IPDPSW.2015.70.
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18.
12. ITU (1999). Recommandation ITU-T P.910. Subjective video quality assessment methods for multimedia applications. Geneva.
13. JCT-VC (2016). HEVC reference software. https://hevc.hhi.fraunhofer.de/.
14. Karczewicz, M., Ye, Y., Chong, I. (2008). Rate distortion optimized quantization. In *VCEG-AH21, Antalya Turkey*.
15. Khan, M.UK., Shafique, M., Henkel, J. (2013). An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding. In *2013 20th IEEE international conference on image processing (ICIP)* (pp. 1578–1582). IEEE.
16. Koivula, A., Viitanen, M., Lemmetti, A., Vanne, J., Hämäläinen, T.D. (2015). Performance evaluation of Kvazaar HEVC intra encoder on Xeon Phi many-core processor. In *2015 IEEE global conference on signal and information processing (GlobalSIP)* (pp. 1250–1254). IEEE.
17. Koivula, A., Viitanen, M., Vanne, J., Hamalainen, T.D., Fasnacht, L. (2015). Parallelization of Kvazaar HEVC intra encoder for multi-core processors. In *2015 IEEE workshop on signal processing systems (SiPS)* (pp. 1–6). IEEE.
18. Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.
19. Lan, C., Xu, J., Sullivan, G.J., Wu, F. (2012). Intra transform skipping. In *JCTVC-I0408, Geneva, CH*.
20. Mercat, A., Arrestier, F., Hamidouche, W., Pelcat, M., Menard, D. (2017). Energy reduction opportunities in an HEVC real-Time encoder. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 1158–1162). IEEE.
21. Mercat, A., Arrestier, F., Pelcat, M., Hamidouche, W., Menard, D. (2017). Prediction of quad-tree partitioning for budgeted energy HEVC encoding. In *2017 IEEE international conference on signal processing systems (SiPS)* (pp. 1–6). IEEE.
22. Mercat, A., Arrestier, F., Pelcat, M., Hamidouche, W., Menard, D. (2018). Machine learning based choice of characteristics for the one-shot determination of the HEVC intra coding tree. In *2018 picture coding symposium (PCS)* (pp. 263–267). IEEE.
23. Mercat, A., Arrestier, F., Pelcat, M., Hamidouche, W., Menard, D. (2018). *Machine learning based choice of characteristics for the one-shot determination of the HEVC intra coding tree* (pp. 263–267). IEEE.
24. MulticoreWare (2017). x265 HEVC Encoder / H.265 Video Codec. http://x265.org/.
25. Peng, K. K., Chiang, J. C., Lie, W. N. (2016). *Low complexity depth intra coding combining fast intra mode and fast CU size decision in 3d-HEVC* (pp. 1126–1130). IEEE.
26. Penny, W., Machado, I., Porto, M., Agostini, L., Zatt, B. (2016). Pareto-based energy control for the HEVC encoder. In *2016 IEEE international conference on image processing (ICIP)* (pp. 814–818). IEEE.
27. Qualcomm (2014). Snapdragon 810 processor product brief. https://www.qualcomm.com/documents/snapdragon-810-processor-product-brief.
28. Quinlan, J.R. (2014). *C4. 5: Programs for machine learning*. Amsterdam: Elsevier.
29. Ruiz, D., Fernández-Escribano, G., Adzic, V., Kalva, H., Martínez, J.L., Cuenca, P. (2015). Fast CU partitioning algorithm for HEVC intra coding using data mining. *Multimedia Tools and Applications*, 861–894. https://doi.org/10.1007/s11042-015-3014-6.
30. Ruiz-Coll, D., Adzic, V., Fernández-Escribano, G., Kalva, H., Martínez, J.L., Cuenca, P. (2014). Fast partitioning algorithm for HEVC Intra frame coding using machine learning. In *2014 IEEE international conference on image processing (ICIP)* (pp. 4112–4116). IEEE.
31. Shen, L., Zhang, Z., An, P. (2013). Fast CU size decision and mode decision algorithm for HEVC intra coding. *IEEE Transactions on Consumer Electronics*, *59*(1), 207–213.
32. Shen, X., & Yu, L. (2013). CU Splitting early termination based on weighted SVM. *EURASIP Journal on Image and Video Processing*, *2013*(1), 4.
33. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T. (2012). Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, *22*(12), 1649–1668. https://doi.org/10.1109/TCSVT.2012.2221191.
34. Sze, V., Budagavi, M., Sullivan, G.J. (Eds.) (2014). *High efficiency video coding (HEVC) integrated circuits and systems*. Cham: Springer.
35. Tan, T.K., Weerakkody, R., Mrak, M., Ramzan, N., Baroncini, V., Ohm, J.R., Sullivan, G.J. (2016). Video quality evaluation methodology and verification testing of HEVC compression performance. *IEEE Transactions on Circuits and Systems for Video*

Technology, *26*(1), 76–90. https://doi.org/10.1109/TCSVT.2015.2477916.
36. UltraVideoGroup (2017). Kvazaar HEVC Encoder. http://ultravideo.cs.tut.fi/#encoder.
37. Vanne, J., Viitanen, M., Hamalainen, T.D., Hallapuro, A. (2012). Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs. *IEEE Transactions on Circuits and Systems for Video Technology*, *22*(12), 1885–1898. https://doi.org/10.1109/TCSVT.2012.2223013.
38. Vantrix (2017). F265 Open Source HEVC/H.265 Project. http://vantrix.com/f-265-2/.
39. Viitanen, M., Koivula, A., Lemmetti, A., Vanne, J., Hamalainen, T. D. (2015). Kvazaar HEVC encoder for efficient intra coding. In *2015 IEEE international symposium on circuits and systems (ISCAS)* (pp. 1662–1665). IEEE.
40. Wang, X., & Xue, Y. (2016). Fast HEVC intra coding algorithm based on Otsu's method and gradient. In *2016 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)* (pp. 1–5). IEEE.
41. Wiegand, T., Sullivan, G., Bjontegaard, G., Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, *13*(7), 560–576. https://doi.org/10.1109/TCSVT.2003.815165.
42. Wien, M. (2015). *High efficiency video coding. signals and communication technology*. Berlin: Springer.
43. Zhang, Y., Kwong, S., Wang, X., Yuan, H., Pan, Z., Xu, L. (2015). Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding. *IEEE Transactions on Image Processing*, *24*(7), 2225–2238. https://doi.org/10.1109/TIP.2015.2417498.
44. Zhang, J., Li, B., Li, H. (2015). An efficient fast mode decision method for inter prediction in HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*, *26*(8), 1502–1515. https://doi.org/10.1109/TCSVT.2015.2461991.

Alexandre Mercat received the Engineering degree in Electronics and Computer Engineering from the National Institute of Applied Sciences (INSA), Rennes in 2015. Now, he is pursuing the Ph.D. degree in Electrical and Computer Engineering from the Institute of Electronics and Telecommunications of Rennes (IETR) laboratory. His research interests include implementation of image and signal processing applications in many core embedded systems, video coding algorithms, complexity-aware video coding, approximate computing, power consumption and digital systems design.



Florian Arrestier is currently pursuing his M.E. degree in Electronics and Computer Engineering at INSA Rennes, France, and his advisor is Prof. Daniel Menard. His current research interests include embedded system, image and signal processing applications and low power design.



Maxime Pelcat is an Associate Professor at the INSA in Rennes. He holds a joint research appointment at IETR in Rennes and at Institut Pascal in Clermont Ferrand, two CNRS research units. Maxime Pelcat obtained his Ph.D. in signal processing from INSA Rennes in 2010, thesis resulting from a collaboration of Texas Instruments, Nice and INSA Rennes. Previously, after one year in the Audio and Multimedia department at Fraunhofer Institute IIS in Erlangen, Germany, he worked as a contractor at France Telecom Research and Development until 2006. He is an author of 50+ peer reviewed publications since 2009 in the domains of models of computation, energy efficiency, multimedia and telecommunication processing, and programming of parallel embedded systems. Maxime Pelcat has served as Guest Editor for the Springer Journal of Signal Processing Systems and is an author of the best paper award at DASIP 2014, the best demo awards at ICME 2015 and EDERC 2014 and of the book "Physical Layer Multi-Core Prototyping" Springer, 2012.

Wassim Hamidouche received the Ph. D. Degree in Signal and Image Processing from the University of Poitiers, France in 2010. From 2011 to 2012 he has been a Research Engineer with Canon Research Centre, Rennes, France. He is Associate Professor at INSA Rennes since 2015 and member of the the Institute of Electronics and Telecommunications of Rennes (IETR), UMR CNRS 6164. His research interests focus on video coding, efficient real time and parallel architectures for the new generation video coding standards, multimedia transmission over heterogeneous networks, and multimedia content security.

Daniel Menard received the Ph.D. and HDR (habilitation to conduct researches) degrees in Signal Processing and Telecommunications from the University of Rennes, respectively in 2002 and 2011. From 2003 to 2012 he was an Associate Professor at the department of Electrical and Computer Engineering (ECE) of the University of Rennes engineering school, ENSSAT. He was also member of the IRISA/INRIA laboratory. He is currently Professor at the ECE department of the engineering school INSA Rennes. He is also member of the IETR/CNRS laboratory. He is author of more than 80 international papers distributed in the areas of embedded systems, video codecs, computer-aided design, arithmetic and signal processing. He is member of the DISPS committee of IEEE Signal Processing society. His research interests include design and implementation of image and signal processing applications in embedded systems, video codecs, approximate computing, fixed-point arithmetic, low power systems.