On predicting the HEVC intra quad-tree partitioning with tunable energy and rate-distortion

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

On Predicting the HEVC Intra Quad-Tree Partitioning with Tunable Energy and Rate-Distortion

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

Abstract Future evolutions of the Internet of Things (IoT) are likely to boost mobile video demand to an unprecedented level. A large number of battery-powered systems will then integrate an High Efficiency Video Coding (HEVC) codec, implementing the latest video encoding standard from MPEG, 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 manage the energy consumption of an HEVC encoder consists in optimizing the quad-tree partitioning and balance compression efficiency and energy consumption. The quad-tree partitioning splits the image into encoding units of variable sizes. The optimal size for a unit is content dependent and affects the encoding efficiency. Finding this optimal repartition is complex and the energy required by the so-called Rate-Distortion Optimization (RDO) process dominates the encoder energy consumption.

For the purpose of budgeting the energy consumption of a real-time HEVC encoder, we propose in this paper a variance-aware quad-tree prediction that limits the energetic cost of the RDO process. The predictor is moreover adjustable by two parameters, $(\underline{\Delta}, \overline{\Delta})$, offering a trade-off between energetic gains and compression efficiency. Experimental results show that the proposed energy reduction scheme is able to reduce the energy consumption of a real-time HEVC encoder by 45% to 62% for a bit rate increase of respectively 0.49% and 3.4%. Moreover, the flexibility offered by parameters $(\underline{\Delta}, \overline{\Delta})$ opens new opportunities for energy-aware encoding management.

Alexandre Mercat, Florian Arrestier, Maxime Pelcat, Wassim Hamidouche, Daniel Menard
INSA Rennes, IETR, UMR CNRS 6164, UEB
20 Avenue des Buttes de Coesmes, 35708 Rennes, France
Tel.: +33-223-238200
Fax: +33-223-238396
E-mail: firstname.lastname@insa-rennes.fr

Maxime Pelcat
Institut Pascal, CNRS UMR 6602, Clermont-Ferrand, France

1 Introduction

With the progress of microelectronics, many embedded applications now encode and stream live video contents. The HEVC [25, 24, 33] standard represents the state-of-the-art in video coding. When compared with the previous ISO/IEC Moving Picture Experts Group (MPEG) Advanced Video Coding (AVC), HEVC Main profile reduces the bit rate by 50% on average for a similar objective video quality [26, 28]. 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 (RD) search process [16].

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 codecs, a solution to reduce energy consumption is to decrease the computational complexity while controlling compression quality losses. By nature, hardware and software encoders differ in terms design choices [12]. This paper puts the focus on software HEVC encoders, however, by reducing the amount of quad-tree configurations to process, this work is promising direction also in the context of hardware encoders.

To reduce the computational complexity of HEVC encoders, several algorithmic solutions have been proposed at the level of quad-tree partitioning, which test less par-

titioning configurations. Exhaustive search solution (optimal) test all possible partitioning configurations and select the one that minimizes the RD-cost. This process is the most time consuming operation in a HEVC encoder and thus it offers the biggest opportunity of complexity reduction (up to 78%) [16]. Complexity reduction solutions at the quad-tree level consist in predicting the adequate level of partitioning that offers the lowest RD-cost. Authors in [23] 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, 7, 31, 11, 19] use CTU texture complexities to predict the quad-tree partitioning. However, these solutions are based on reducing the complexity of the best effort (i.e. non-real-time) HEVC test Model (HM) encoder and do not offer the possibility to tune compression quality and complexity. Moreover, a real-time encoder such as Kvazaar is up to 10 times faster than HM [30]. The performance of state-of-the-art solutions based on HM are biased because they are measured comparatively to a gigantic compression time. The complexity overhead of state of the art solutions is thus comparatively higher in the context of a real-time encoder. Published results can thus not be directly applied to reduce the energy consumption in a real-time encoder without knowledge of complexity overhead.

This paper proposes a new lightweight method to predict the CTU quad-tree partitioning with the objective to budget the energy consumption of a real-time HEVC encoder. The proposed approach is used to limit the recursive RDO process, which drastically impacts the energy consumption of HEVC Intra encoder [16]. Compared to our previous work [17] and existing methods, the predictor is also made adjustable, based on two parameters that provide a trade-off between energy consumption and coding efficiency. The proposed energy reduction technique exploits the correlation between the CTU partitioning and the variance of the Coding Unit (CU) luminance samples. Compared to state-of-the-art solutions, the originality of the presented work comes from its focus on energy consumption, real-time encoders and adjustable energy gains targeted towards energy budgeting.

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 proposed algorithm of quad-tree partitioning prediction based on variance studies. The proposed energy reduction scheme and its performance are investigated in Section 4. Finally, Section 5 concludes the paper.

2 Related works

2.1 HEVC Encoding and its Rate Distorsion Optimisation

An HEVC encoder is based on a classical *hybrid video encoder* structure that combines Inter-images and Intra-image predictions. The HEVC standard defines *units* that refer to parts of the produced bitstream, and *blocks* that refer to parts of the images to encode. While encoding in HEVC, each frame is split into equally-sized Coding Tree Units (CTUs) (Figure 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 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 Figure 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 [24]. Each mode correspond 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 with:

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

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.

2.2 Studied HEVC Encoder

For embedded applications, hardware encoding solutions [21] consume much lower energy than software solutions. However, when the considered system does not embed a hardware coprocessor, a software HEVC encoder [9, 18, 29, 27] can be used, including 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

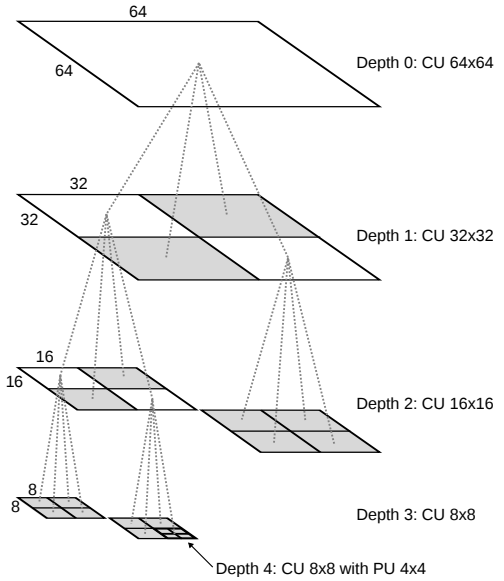

Fig. 1 Quad-tree structure of a Coding Tree Unit (CTU), divided into Coding Units (CUs) and Prediction Units (PUs) (dimensions in luminance pixels).

complexity of HM is high and not adapted to embedded applications. To fill this gap, the *x265*, *f265* and *Kvaazar* 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 [27] for its real-time encoding capacity. The conclusions of this study can however be extended to the other real time software or hardware encoders, as they all rely on a classical RDO process to reach acceptable compression performance.

2.3 Complexity Reduction of the Quad-Tree Partitioning

As shown in [16], 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 processing when future 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 [23, 34, 4] 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%. We reimplemented in the real-time software encoder Kvazaar two complexity reduction techniques extracted from [23] and [34]. For the algorithm from [23], we obtain with our implementation an average of energy reduction of 37.9% for a BD-BR increase of 0.54%. Using [34], our results show that the complexity reduction technique reduces the energy consumption of 30% in average for a BD-BR increase of 0.12%.

Authors of [22] present 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 features. This algorithm reduces the coding time by 52% at the expense of bit rate increase of 2%.

Works in [1, 7, 31, 11, 19] 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% for a BD-BR increase of 0.8%. Feng et al. [7] use information entropy of CUs and sub-CUs saliency maps to predict the CUs size. The method reduces the complexity of 37.9% for a BD-BR increase of 0.62%.

Khan et al. [11] 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% 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 proposed method, while based on variance, does not consider the Rayleigh distribution and thus differs from [11].

In [20], Penny et al. propose the first Pareto-based energy controller for an HEVC encoder. From [20] 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%, authors achieve a BD-BR increase between 6.84% and 25%, respectively.

Except [20], none of the presented works offer features to adapt quality and complexity. Including [20], these studies are all based on complexity reduction of the HM software encoder and their performances do not translate to real-time encoders.

3 CTU Variance Analysis for Predicting an HEVC Quad-Tree Partitioning

The aim of this paper is to replace the brute force scheme usually employed in HEVC encoders by a low complexity algorithm which predicts 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 whether a CU at a current depth $d \in [1, 4]$ should be encoded or need to be further merged into lower depth $d - 1$.

It has been already 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 [11, 1, 31]. Figure 2 shows the CU boundaries of the 6th frame of *BasketballDrive* video sequence. It is noteworthy that the regions with the lowest variance (smooth) tend to be encoded with larger blocks, as illustrated by the green circle in Figure 2, while the blue circle shows a region with a high variance (high local activity), which are encoded with smaller blocks. This existing correlation between the pixel's values of a block (variance) and its encoding decomposition can be used to predict the quad-tree decomposition of a CTU and thus reduce drastically the encoding complexity.

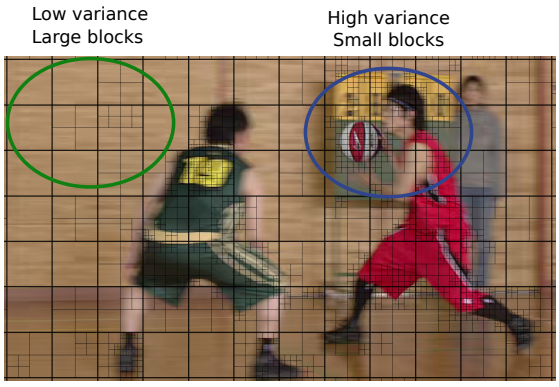

Fig. 2 Quad-tree partitioning of the 6th HEVC intra coded frame of the *BasketballDrive* sequence. The green (resp. blue) circle shows that the lowest (resp. highest) variance regions tend to be encoded with larger (resp. smaller) units.

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 $d' < d$. However, 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.

In the following paragraphs, two approaches are investigated to classify CUs in one of these two populations based on the variance criteria.

3.1.1 PDF approach

Figure 3 shows the PDFs of the variance of the CU 8×8 of the two populations for the 6th frame of *BasketballDrive* video sequence. Statistically, for two PDFs with sufficiently distinct populations, a threshold can be derived such that, given an element — here the variance of a 8×8 CU — the element can be classified to one of the two populations. The classification threshold is given by the abscissa of the intersection of the two PDFs, represented by the green dotted line in Figure 3. The threshold is determined such as to minimize the miss detection and false alarm probability.

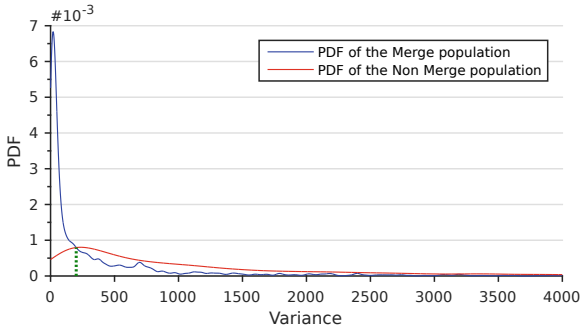

Fig. 3 Probability Density Functions (PDFs) of CUs 8×8 variances of the two populations *Merged* and *Not Merged* of the sixth frame of the sequence *BasketballDrive*. A variance threshold (the green dotted line) can be used to classify a block into one of the two populations.

3.1.2 Cumulative Distribution Function (CDF) approach

An alternative approach consists in using only the CDF of the *Non Merged* population 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 terms, it is rare for a CU to have a variance greater than a certain threshold. From this observation, a variance threshold $v_{th}(\Delta, d)$ for each depth d can be extracted from the inverse CDF curve for a specific probability Δ. For example, Figure 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 Figure 4. Δ is the percentage of

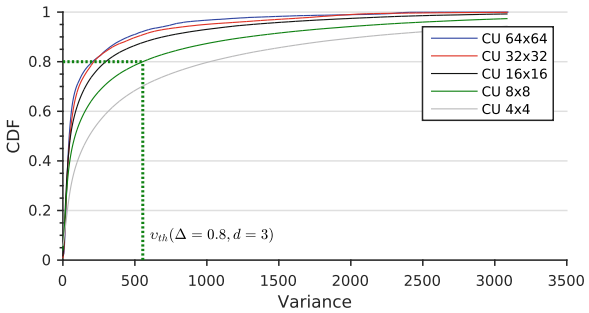

Fig. 4 CDFs of the Merged population depending of CU size for the sixth frame of the sequence *BasketballDrive*. Under a specific probability Δ, a variance threshold can be extracted from the inverse CDF curve to classify a block as Merged.

coding units whose variance is under the threshold v_{th}, i.e. the variance threshold that triggers unit split.

Table 1 Variance thresholds $v_{th}(\Delta, d)$ 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

Table 1 shows the thresholds $v_{th}(\Delta, d)$ for $d \in \{1, 2, 3, 4\}$ extracted from the CDFs using the second previous approach for the 50th frame of the two sequence *PeopleOnStreet* and *ParkScene*. The Table illustrates the large variation of the threshold value across different video contents. In fact, the thresholds rely on the video contents and thus have to be determined on-the-fly from the 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 frames. The modelling of these thresholds can be conducted using variance PDFs with an approximation of the distribution based on common known probability distribution or directly using population features. Neither of those two approaches were conclusive in our experimental results.

However, an approximation of the thresholds directly from *Non Merged* population features provides good results for the CDF curve. In Figure 5, the variance thresholds $v_{th}(d = 3, \Delta = 0.8)$ is plotted versus the mean $\mu_v(d = 3)$ 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, $v_{th}(\Delta, d)$ depends linearly on $\mu_v(d)$ and standard deviation $\sigma_v(d)$, and this independently of the QP value.

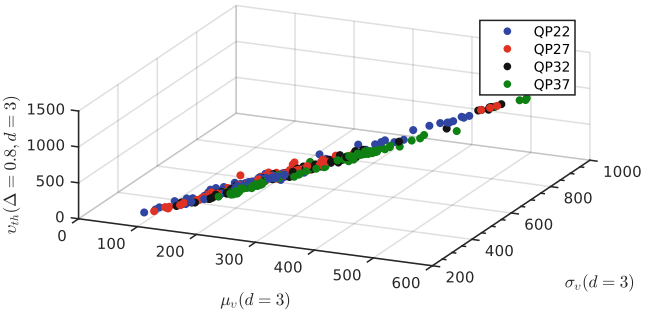

Fig. 5 Variance thresholds $v_{th}(d = 3, \Delta = 0.8)$ versus the mean $\mu_v(d = 3)$ and the standard deviation $\sigma_V(d = 3)$ of CUs 8×8 variances. For a fixed value of Δ, the variance threshold values form a plane (independently of the QP value).

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

$$v_{th}(\Delta, d) = a(\Delta, d) \cdot \mu_v(d) + b(\Delta, d) \cdot \sigma_v(d) + c(\Delta, d) \quad (2)$$

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 Coefficient of Determination (Rsq) is a metric that quantifies the accuracy of a predicting model. It falls between 0 and 1 and the more the Rsq is close to 1, the more the predicted data fits the actual data. Rsq is defined by Equation (3) where $\hat{y}$ represents the estimated values of y, n is the number of y value used for the measure and $\bar{y}$ is the mean of y.

$$Rsq = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (3)$$

Table 2 summarizes the average of the Rsq for $\Delta \in \{0.3, 0.35, \ldots, 0.9\}$ for each depth d. The average Rsq value is ever larger than 72%, which confirms that the model fits the $v_{th}(\Delta, d)$, regardless of the video content and QP value.
Summarizing the above analysis:

- Thresholds from CDFs of variances can be predicted from reference Learning Frames (F_L).
- Look-Up Tables (LUTs) requires slight computation and memory overhead for the determination of the threshold. The computation overhead of the proposed

Table 2 *Rsq* average of $V_{th}(\Delta, d)$ modelling

Depth	Average *Rsq*
$d = 0$	0.8774
$d = 1$	0.8482
$d = 2$	0.7214
$d = 3$	0.8942
$d = 4$	0.9436

method presented in section 4 is between 1% and 1.9% depending on the QP and video sequence.
- The prediction of thresholds is independent from the QP value (Figure 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 the proposed algorithm to predict the CTU partitioning using a variance criterion and the thresholds $v_{th}(\Delta, d)$.

3.3 Prediction Algorithm for 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.

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) Cdm matrix of the CTU partitioning of Figure 1. Each element of the matrix represents the depth of an 8 by 8 pixel square in the CTU.

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) Rcdm matrices of the Cdm illustrated Figure 6a. In grey are repsented the blocks merged by the Rcdm algorithm.

Fig. 6 Cdm and its associate Rcdm matrices of the CTU partitioning of Figure 1

For a given CTU, let $v^d(i, j)$ 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 Figure 7.

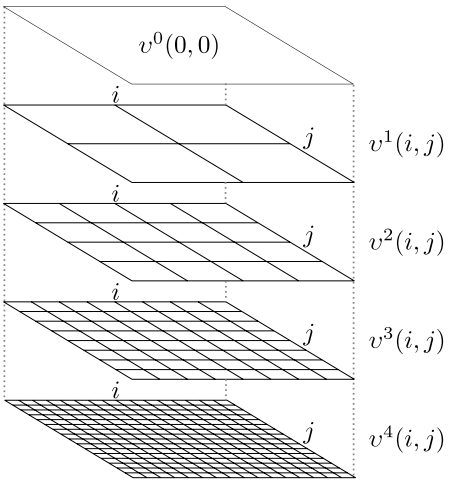

Fig. 7 $v^d(i, j)$: variance of the luminance sample blocks of size $2^{6-d} \times 2^{6-d}$ of a CTU versus the depth level d.

Algorithm (1) describes our proposed algorithm that predicts the CTU partitioning. The algorithm takes as inputs the luminance samples of CTU and the table of thresholds V_{th} previously computed by Equation (2) to generate the Cdm associated to the input CTU. In other terms, the goal of this algorithm is to use the variances of the luminance samples to determine the Cdm matrix of the CTU. Then, the encoder only uses the predicted depths instead of RDO to encode the video, reducing significantly the complexity.

First of all, the full Cdm is initialized with the depth value 4 (line 1) and all the variances $v^4(x, y)$ of the CU 4×4 (line 2) are computed using Equation (4) 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.

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

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. 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 9) previously detailed in Section 3.1. If the 4 blocks variances v^d are

Algorithm 1: CDM_Generator

Data: Samples of CTU, $V_{th}(\Delta, d)$
Result: CDM matrix

1 $\text{CDM}(x, y) = 4, \forall x, y \in [0, 7]$; // Initialization
2 Compute: $v^4(x, y), \forall x, y \in [0, 15]$; // cf Eq. 4
3 **for** $(d = 4; d > 0; d--)$ **do**
4 $\quad \delta = \max(2^{d-2}, 1)$; // CDM matrix block size
5 $\quad$ **for** $(y = 0; y < 8; y \mathrel{+}= \delta)$ **do**
6 $\quad\quad$ **for** $(x = 0; x < 8; x \mathrel{+}= \delta)$ **do**
$\quad\quad\quad$ // Test if the 4 neighbor blocks have the same depth d when $d < 4$
7 $\quad\quad\quad$ **if** $((\text{CDM}(x, y) = d \;\&\&$
$\quad\quad\quad$ $\text{CDM}(x + \delta, y) = d \;\&\&$
$\quad\quad\quad$ $\text{CDM}(x, y + \delta) = d \;\&\&$
$\quad\quad\quad$ $\text{CDM}(x + \delta, y + \delta) = d) \quad || \quad (d = 4))$
$\quad\quad\quad$ **then**
$\quad\quad\quad\quad$ // Test if the variances of blocks are lower than the threshold
8 $\quad\quad\quad\quad$ $x' = x/(2^{3-d}); y' = y/(2^{3-d})$; // Var idx
9 $\quad\quad\quad\quad$ **if** $(v^d(x', y') < V_{th}(\Delta, d) \;\&\&$
$\quad\quad\quad\quad$ $v^d(x' + 1, y') < V_{th}(\Delta, d) \;\&\&$
$\quad\quad\quad\quad$ $v^d(x', y' + 1) < V_{th}(\Delta, d) \;\&\&$
$\quad\quad\quad\quad$ $v^d(x' + 1, y' + 1) < V_{th}(\Delta, d))$ **then**
$\quad\quad\quad\quad\quad$ // Block merging in the CDM
10 $\quad\quad\quad\quad\quad$ $\text{CDM}(x' + i, y' + j) = d - 1,$
$\quad\quad\quad\quad\quad$ $\forall i, j \in [0, \delta - 1]$;
11 $\quad\quad\quad\quad\quad$ Compute: $v^d(x'/2, y'/2)$; // cf Eq. 5

lower than the given threshold $V_{th}(\Delta, d)$ then the blocks are merged and the corresponding elements in the CDM are set to $d - 1$ (line 10) and the variance of the merged block is calculated three times using the combined variance Equation 5.

$$v_{a \cup b} = \frac{(2n - 1)(v_a + v_b) + n(\mu_a - \mu_b)^2}{4n - 1} \tag{5}$$

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

3.4 Relationship between the Prediction Performance and the Δ Parameter

As described in Section 3.3, Algorithm (1) takes as input a set of variance thresholds v_{th} corresponding to the parameter Δ. As detailed in Section 3.1, the parameter Δ is the normalized probability of *Non Merged* population with a variance less or equal to its associated threshold $v_{th}(\Delta, d)$. The following section is a baseline study of the impact of parameter Δ.

3.4.1 The CDM Distance for Performance Analysis

The main objective of Algorithm (1) 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:

$$\Gamma(A, B) = \left[\sum_{i=0}^{7} \sum_{j=0}^{7} |A(i, j) - B(i, j)| \right] /64 \tag{6}$$

where A and B are the two compared CDMs. In other terms, 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 Figures 6 as example, the distance Γ between the CDM Figure 6a and Figure 6b is equal to $\Gamma = (4 + 1 + 16)/64 = 0.3281$. Distance Γ is used in the following sections to evaluate the accuracy of the prediction.

3.4.2 CDM Distance Versus Δ

In Figure 8 are plotted in blue the average and the standard deviation of $\Gamma_\Delta(P, R)$ where $\Gamma_\Delta(P, R)$ is the distance between the predicted CDM P and the reference CDM R [1], generated by a full RDO process (optimal) according to $\Delta \in \{0.3, 0.35, ..., 0.95\}$. The results are extracted from the 100 first frames of the 5 following sequences: *BasketballDrive*, *BQTerrace*, *Cactus*, *ParkScene*, *PeopleOnStreet* and *Traffic* for 4 QP values: 22, 27, 32 and 37. The more $\Gamma(P, R)$ is close to 0, the more precise the predicted CDM P becomes.

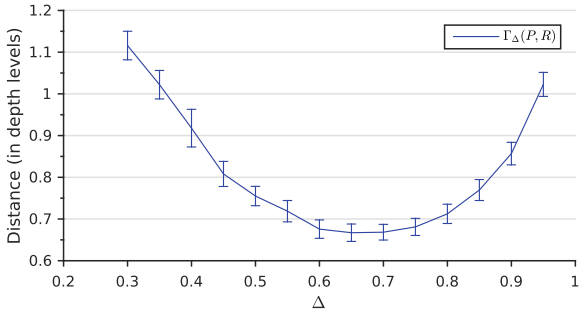

Fig. 8 Averages and standard deviations of $\Gamma_\Delta(P, R)$, the distance between the predicted CDMs P and the reference CDM R generated by a full RDO process versus Δ. The distance is minimized, i.e. the predictions accuracy is maximized for $\Delta \in [0.6, 0.7]$.

Figure 8 shows that $\Gamma_\Delta(P, R)$ has the property of being convex when plotted over Δ. Furthermore, the low standard deviations values (< 0.45) of $\Gamma_\Delta(P, R)$ (represented by the vertical bars in Figure 8) show that the results are stable across the video contents and QP value.

[1] exhaustive search leading to the optimal solution

As can be seen in Figure 8, $\Gamma_{\Delta}(P,R)$ is minimized and thus the accuracy of the predictions is maximized for $\Delta \in [0.6, 0.7]$. Moreover, the value of $\Gamma_{\Delta}(P,R)$ is below 0.7 which demonstrates the accuracy of the proposed predictions.

3.4.3 Decomposing the CDM *Distance into Lower and Upper Distances*

To extend the previous analysis, the distance $\Gamma(A,B)$ can be decomposed into two independent distances: $\overline{\Gamma(A,B)}$ and $\underline{\Gamma(A,B)}$ complying with Equation (7):

$$\Gamma(A,B) = \overline{\Gamma(A,B)} + \underline{\Gamma(A,B)} \tag{7}$$

where $\overline{\Gamma(A,B)}$ (respectively $\underline{\Gamma(A,B)}$) is called *upper distance* (respectively *lower distance*) and is the normalized distance between the two CDMs A and B only when the depth of A is lower (respectively higher) than the depth of B, as described by Equation (8) (respectively Equation (9)). $\overline{\Gamma(A,B)}$ is called the *upper distance* as it represents the fact that the quad-tree decomposition of CDM A is shallower than the one of B, i.e the CDM of A has lower depth values. The same reasoning can be applied to $\underline{\Gamma(A,B)}$.

$$\overline{\Gamma(A,B)} = \left[\sum_{i=0}^{7} \sum_{j=0}^{7} |\min\left(A(i,j) - B(i,j), 0\right)| \right] /64 \tag{8}$$

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

In green and red in Figure 9 are plotted the averages and the standard deviations of respectively $\overline{\Gamma_{\Delta}(P,R)}$ and $\underline{\Gamma_{\Delta}(P,R)}$, i.e. the *upper distance* and *lower distance* between the predicted CDM P and the reference CDM R generated by a full RDO process with $\Delta \in \{0.3, 0.35, ..., 0.95\}$. The decomposition of $\Gamma_{\Delta}(P,R)$ into $\overline{\Gamma_{\Delta}(P,R)}$ and $\underline{\Gamma_{\Delta}(P,R)}$ is interesting as it tells the difference between the two following types of prediction errors: the error when the predicted depth is lower than the optimal depth and when the predicted depth is higher than the optimal depth. Figure 9 shows that $\overline{\Gamma_{\Delta}(P,R)}$ and $\underline{\Gamma_{\Delta}(P,R)}$ are respectively strictly decreasing and increasing when Δ increases.

For a low value of Δ, such as $\Delta = 0.3$ in Figure 9, the main part of errors is due to low values of depth in the predicted CDM P, i.e. $\underline{\Gamma_{\Delta}(P,R)}$ close to 0. Indeed, as shown by Figure 4, a low value of Δ implies low values of thresholds $v_{th}(\Delta, d)$ which according to Algorithm (1) lead to fewer merges during the CDM generation. The resulting CDM has therefore a finer-grained splitting and minimizes the prediction errors on small

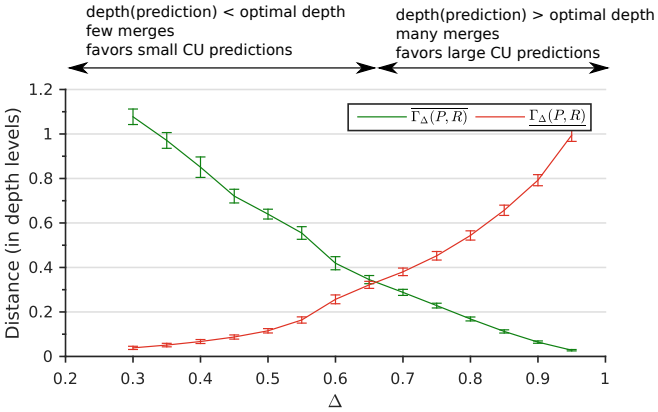

Fig. 9 Averages and standard deviations of $\overline{\Gamma_{\Delta}(P,R)}$ and $\underline{\Gamma_{\Delta}(P,R)}$, the *upper* and *lower distance* between the predicted CDM P and the reference CDM R generated by a full RDO process versus Δ.

CUs in the CTU. In contrast, for a high value of Δ, such as $\Delta = 0.95$ in Figure 9, the resulting CDM has a coarser-grained splitting and minimizes the prediction errors of large CUs in the CTU.

As a result of the previous analysis, we propose to force the HEVC encoder to encode between two CDMs: $\text{CDM}_{\overline{\Delta}}$ and $\text{CDM}_{\underline{\Delta}}$ generated from two different values of Δ with $\overline{\Delta} < \underline{\Delta}$ to minimize both errors $\overline{\Gamma_{\Delta}(P,R)}$ and $\underline{\Gamma_{\Delta}(P,R)}$.

3.5 Refinement Algorithm for CTU partitioning

To increase the accuracy of the 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 Refined CTU Depth Map (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 is to check whether the input CDM's 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 are of 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 (Figure 6a) and its associated RCDM (Figure 6b) matrices. The grey blocks in the RCDM Figure 6b represent the merged

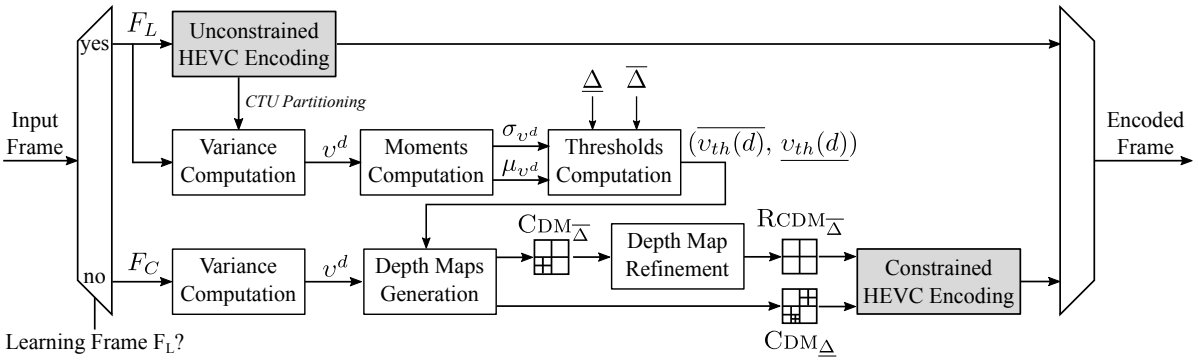

Fig. 10 Diagram of the proposed method. The Learning Frames (F_L) are encoded with a full RDO process (unconstrained) and the block variances of the resulting quad-tree decomposition is used to compute the set of thresholds $\underline{v_{th}(d)}$ and $\overline{v_{th}(d)}$. The constrained frames (F_C) use these thresholds to generate two CDMs and constrain the encoder to only apply the RDO process in the interval formed by the two CDMs.

Algorithm 2: CDM_Refinement

Data: CDM matrix
Result: RCDM matrix
if $\text{CDM}(0,0) = 0$ **then**
  $\text{RCDM}(x,y) = 0, \forall x,y \in [0,7]$;// Set RCDM at 0
else
  for $(y = 0; y < 8; y++)$ **do**
    for $(x = 0; x < 8; x++)$ **do**
      if $\text{CDM}(x,y) = 4$ **then**
        $\text{RCDM}(x,y) = 3$ // Automatic merge
      else
        $d = \text{CDM}(x,y)$ // Get the depth
        $N = 16/2^d$ // Offset definition
        // Test if it is the last blocks in the Z-scane order for the depth d
        if $((x \,\%\, N) = \frac{N}{2})$ && $((y \,\%\, N) = \frac{N}{2})$ **then**
          // Test if the three other blocks have the same depth
          if $(\text{CDM}(x - \frac{N}{2}, y) = d$ && $\text{CDM}(x, y - \frac{N}{2}) = d$ && $\text{CDM}(x - \frac{N}{2}, y - \frac{N}{2}) = d)$ **then**
            $\text{RCDM}(x+i, y+j) = d, \forall i,j \in [0, \frac{N}{2}]$; // Fill the RCDM

blocks. The next section describes our proposed energy reduction scheme.

4 Using Quad-Tree Partitioning Prediction to Reduce the Encoder Energy Consumption

To reduce the energy consumption of HEVC encoder, we propose to limit the exhaustive search of the RDO process on the CTU quad-tree decomposition. Our proposed energy reduction technique aims to predict the coding-tree partitioning from video frame content properties. We propose a variance-aware quad-tree partitioning prediction.

To limit the exploration of the quad-tree decomposition of a CTU, two CDMs, i.e. *refined upper quad-tree constraints* ($\text{RCDM}_{\overline{\Delta}}$) and *lower quad-tree constraints* ($\text{CDM}_{\underline{\Delta}}$), are predicted from the variance of its samples before starting any encoding computation. Then, the HEVC encoder is forced to only apply the RDO process between the two CDMs.

4.1 Overall Algorithm Scheme

Figure 10 presents a high-level diagram of our overall algorithm scheme. Firstly, the video sequence is split into Groups of Frames (GOF).

The first frame of each GOF, called Learning Frame (F_L) is encoded with a full RDO process. From this encoding are extracted the variances v^d for each 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_{v^d} and the standard deviations σ_{v^d} of the variance populations v^d. Based on two parameters $\underline{\Delta}$ and $\overline{\Delta}$, two sets of thresholds $\underline{v_{th}(d)}$ and $\overline{v_{th}(d)}$ are calculated using Equation (2) and the LUT of the coefficients $a(\Delta,d)$, $b(\Delta,d)$ and $c(\Delta,d)$ computed off-line (cf. Section 3.2).

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 twice using the two sets of thresholds previously computed from frame F_L. Algorithm (1) takes as input the set of thresholds $\underline{v_{th}(d)}$ the first time and $\overline{v_{th}(d)}$ the second one to generate respectively the $\text{CDM}_{\underline{\Delta}}$ and $\text{CDM}_{\overline{\Delta}}$. To finish, the $\text{CDM}_{\overline{\Delta}}$, being the *upper quad-tree constraint*, is refined by Algorithm (2) to build $\text{RCDM}_{\overline{\Delta}}$. This refinement pro-

cess gives more slack to the constrained depth decision that will test coarser grain units.

To conclude this section, our proposed energy reduction scheme takes as input two parameters $\overline{\Delta}$ and $\underline{\Delta}$ with $\overline{\Delta} < \underline{\Delta}$ to generate the two CDMs: $\text{CDM}_{\overline{\Delta}}$ and $\text{CDM}_{\underline{\Delta}}$ predicted from the variance of its samples. Then, the $\text{CDM}_{\overline{\Delta}}$ is refined to generate the $\text{RCDM}_{\overline{\Delta}}$. The $\text{CDM}_{\underline{\Delta}}$ and $\text{RCDM}_{\overline{\Delta}}$ constitute a smart interval of quad-tree partitioning. The HEVC encoder is then forced to only apply the RDO process in the interval between $\text{CDM}_{\underline{\Delta}}$ and $\text{RCDM}_{\overline{\Delta}}$.

4.2 Experimental Setup and Results

The following section gives the experimental setup and the results obtained for the proposed energy reduction scheme on a real time HEVC encoder.

4.2.1 Experimental Setup

To conduct the experiments, 17 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 [13, 14, 30] in All Intra (AI) configuration. Since the configuration aims to be real-time, from [16], the Rate-Distortion Optimisation Quantization (RDOQ) [10] and the Intra transform skipping [15] features are disabled. Each sequences is encoded with 4 different QP values: 22, 27, 32, 37 [2].

Bjøntegaard Delta Bit Rate (BD-BR) and Bjøntegaard Delta PSNR (BD-PSNR) [32] 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: Peak Signal-to-Noise Ratio (PSNR). Similarly, the BD-PSNR measure 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 [8], RAPL power measurements are coherent with external measurements and [6] 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.

4.2.2 Experimental Metrics and Parameters

The performance of the proposed energy reduction scheme is evaluated by measuring the trade-off between Energy Reduction (ER) in % and RD efficiency using the BD-BR and BD-PSNR. ER is defined by Equation (10):

$$\text{ER} = \frac{100}{4} \sum_{QP_i \in \{22,27,32,37\}} \frac{E_{Ref}(\text{QP}_i) - E_{red}(\text{QP}_i)}{E_{Ref}(\text{QP}_i)} \tag{10}$$

where $E_{Ref}(\text{QP}_i)$ is the energy spent to encode the video sequence without constraint and $E_{red}(\text{QP}_i)$ the energy to encode the same sequence with our proposed energy reduction scheme, both with $\text{QP} = \text{QP}_i$.

Table 3 Configuration of $(\underline{\Delta}, \overline{\Delta})$

Configuration	$(\underline{\Delta}, \overline{\Delta})$
C1	(0.90,0.30)
C2	(0.85,0.35)
C3	(0.80,0.40)
C4	(0.75,0.45)
C5	(0.70,0.50)
C6	(0.65,0.55)
C7	(0.60,0.60)

Table 3 summarizes the configurations of $(\underline{\Delta}, \overline{\Delta})$ used to evaluate the performance of the proposed energy reduction scheme. We vary the couple of parameters $(\underline{\Delta}, \overline{\Delta})$, starting from the base value $(\underline{\Delta}, \overline{\Delta}) = (0.60, 0.60)$ (C7 configuration), corresponding approximately to the crossing point of curves in Figure 9. As the error is close to symmetric, we keep the sum $\underline{\Delta} + \overline{\Delta}$ constant until $(\underline{\Delta}, \overline{\Delta}) = (0.30, 0.90)$ (C1 configuration), with incremental steps of 0.05.

As detailed in Section 4, the proposed reduction scheme is composed by two distinct algorithms: CDM *Generator* (Algorithm (1)) and CDM *Refinement* (Algorithm (2)). To evaluate the gain of the second one, each encoding was carried out with and without applying the CDM *Refinement* algorithm. In other words, the HEVC encoder is forced to only apply the RDO process between $\text{CDM}_{\underline{\Delta}}$ and $\text{CDM}_{\overline{\Delta}}$ (respectively $\text{RCDM}_{\overline{\Delta}}$) when the scheme does not apply (respectively apply) the CDM *Refinement* algorithm (see Figure 10).

4.2.3 Experimental Results

Figures 11 and 12 show the results in term of BD-BR and BD-PSNR versus ER of the configurations detailed in Table 3 when the CDM *Refinement* algorithm is and is not applied. The energy values include the energy overhead due to the entire energy reduction scheme as the variance computation. The results of Figures 11 and 12 are the average of the results obtained when applying our proposed energy reduction scheme on the 17 video sequences [2]. Moreover, Table 4 details the results for the 17 different sequences belonging to 5 classes (A, B, C, D and E) for configurations C7 and C1 (when the RCDM algorithm is applied).

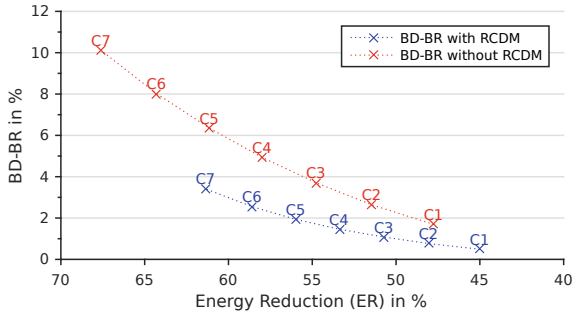

Fig. 11 BD-BR and ER versus $(\underline{\Delta}, \overline{\Delta})$ configurations summarized in Table 3.

First of all, the results of Figures 11 and 12 show that applying the CDM *Refinement* algorithm consumes between 2.1% (C1) and 4.8% (C7) more energy with not negligible gains of encoding performance. The gains obtained by applying the CDM *Refinement* algorithm are between:

- 1.2% (C1) and 6.7% (C7) in term of BD-BR,
- 0.06 dB (C1) and 0.33 dB (C7) in term of BD-PSNR.

These results confirm the interest of the CDM *Refinement* algorithm, thus the rest of this section is focused only on the results with the CDM *Refinement* algorithm.

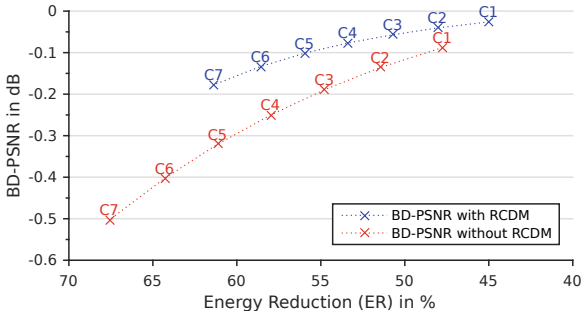

Fig. 12 BD-PSNR and ER versus $(\underline{\Delta}, \overline{\Delta})$ configurations summarized in Table 3.

The results show that our energy reduction scheme is able to reduce the energy consumption of the HEVC encoder from 45% (C1) to 62% (C7). In configuration C7, an average of 62% of energy reduction is achieved with a 3.4% BD-BR increase and a quality degradation of -0.177dB BD-PSNR. Whereas in configuration C1, an average of 45% of energy reduction is achieved with a slight BD-BR increase of 0.49% and a quality degradation of -0.03dB BD-PSNR

Table 4 shows that for the two configuration C7 and C1, the ER has a range lower than $\approx 6\%$ across the sequences with a maximum of 63.76% in *RaceHorses* and a minimum of 58.12% in *BlowingBubbles* for configuration C7 for example. The variations of ER for all configurations (from C1 to C7), are due to the number of blocks merged to build the $\text{RCDM}_{\overline{\Delta}}$, which depends on the prediction of the $\text{CDM}_{\overline{\Delta}}$. This decomposition depends on the texture characteristics of the sequence. This instability is especially significant for configuration C7 because this configuration has the most severe constraint, i.e. the $\text{CDM}_{\underline{\Delta}}$ and $\text{CDM}_{\overline{\Delta}}$ are the same (generated by the same Δ value: 0.6). Only the CDM *Refinement* algorithm creates a small unpredictable interval between the two CDMs.

In the other hand, it can be noticed that classes C and D for the C7 and C1 configurations have less degradation in terms of BD-BR and BD-PSNR than other classes. The same results are observed for all configurations (from C1 to C7). It can be explained by the condition line 7 of Algorithm (1) and the small resolution of these two classes which tend to have a finer-grained splitting during the RDO process. Indeed, Algorithm (1) does not try to merge the block at the current depth if the four neighbor blocks in the Z-scan order do not have the same depth that tends to predict more finer-grained splitting.

4.2.4 Comparison with Related Works

Figure 13 presents a comparison between our proposed energy reduction technique and techniques extracted from the literature (Section 2). This comparison is unfair to our method in terms of time/energy, as the complexity overhead of the proposed complexity reduction techniques in literature would be higher in the context of a real time encoder, thus reducing their complexity reduction. In a real-time configuration (see 4.2.1), the computational overhead of our proposed method in the real-time encoder Kvazaar is between 1% and 1.9%. Results from techniques implemented of HM [22, 4, 11, 7, 1, 20] and real-time encoder [23, 34] are plotted in red and blue, respectively. Crosses and circles correspond respectively to results in term of energy and complexity reductions.

Figure 13 shows that, even with an unfair comparison, the energy reduction technique proposed in this paper outperforms state of the art both in terms of average encoding energy/time reduction and compression efficiency across the different configurations (from C1 to C7). In configuration C1 for example, our method reduces the energy consumption by up to 7% more when compared to [23] for the same bit-rate degradation of

Table 4 BD-BR, BD-PSNR and ER of the proposed energy reduction scheme according to the sequences for configurations C7 and C1

				C7 configuration			C1 configuration		
Class	**Sequence name**	**Resolution**	**Nb Frame**	**BD-BR** (in %)	**BD-Psnr** (in dB)	**ER** (in %)	**BD-BR** (in %)	**BD-Psnr** (in dB)	**ER** (in %)
A	*Traffic*	3840x2048	150	4.59	-0.24	63.11	0.97	-0.05	45.58
A	*PeopleOnStreet*	2560x1600	150	4.28	-0.24	62.12	0.63	-0.04	47.47
B	*ParkScene*	1920x1080	240	4.29	-0.19	58.79	0.92	-0.04	40.67
B	*Cactus*	1920x1080	500	3.71	-0.11	63.53	0.52	-0.02	45.71
B	*BQTerrace*	1920x1080	600	3.56	-0.13	63.19	0.46	-0.02	45.03
B	*BasketballDrive*	1920x1080	500	2.26	-0.15	63.76	0.16	-0.03	48.66
C	*RaceHorses*	832x480	300	3.12	-0.18	62.60	0.42	-0.02	44.47
C	*BQMall*	832x480	600	1.88	-0.13	61.78	0.12	-0.01	47.42
C	*PartyScene*	832x480	500	2.74	-0.13	61.65	0.04	-0.00	42.96
C	*BasketballDrill*	832x480	500	3.46	-0.19	60.98	0.44	-0.02	45.55
D	*RaceHorses*	416x240	300	2.47	-0.15	61.75	0.14	-0.01	40.88
D	*BQSquare*	416x240	600	2.74	-0.21	60.62	0.38	-0.03	50.44
D	*BlowingBubbles*	416x240	500	1.45	-0.10	58.12	0.03	-0.00	41.59
D	*BasketballPass*	416x240	500	2.39	-0.14	62.42	0.15	-0.01	43.85
E	*FourPeople*	1280x720	600	4.79	-0.27	59.57	0.99	-0.06	45.71
E	*Johnny*	1280x720	600	6.02	-0.24	59.24	1.37	-0.06	41.68
E	*KristenAndSara*	1280x720	600	4.15	-0.21	59.73	0.62	-0.03	47.25
			Average	**3.41**	**-0.18**	**61.35**	**0.49**	**-0.03**	**45.00**

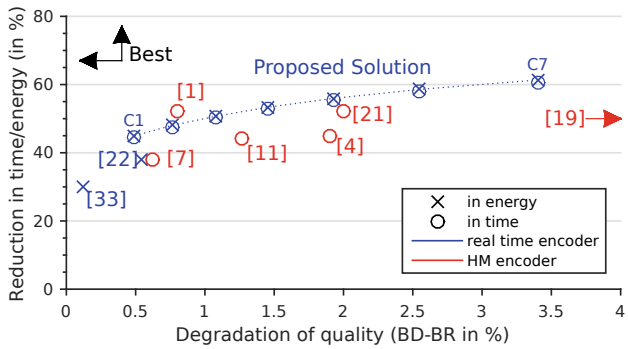

Fig. 13 Comparison with related works in term of time/energy reduction (in %) and quality degradation (BD-BR in %).

$\approx 0.5\%$ of BD-BR. Regarding [34], our results show that the complexity reduction technique reduces the energy consumption of 30% in average for a BD-BR increase of 0.12%. An energy reduction of 30% cannot be obtained with our current configurations of $(\underline{\Delta}, \overline{\Delta})$. Only [1] has better results than our configurations. This technique uses 8 image filtering to compute their local and global edge complexities and predict the partitioning of CUs. We reimplemented in the real-time software encoder Kvazaar only the filtering to estimate the overhead of this complexity reduction technique. We do not have optimized all the computation but the results show that the overhead of the filtering is between 20% and 30% for the real-time configuration describe in Section 4.2.1 depending of the QP value. This overhead would lead to a decrease of the energy/time reduction under our results.

5 Conclusion

This paper presents an energy reduction scheme for HEVC encoders based on a CTU partitioning prediction technique that limits the recursive RDO process. The proposed energy reduction technique exploits the correlation between a CTU partitioning, the texture complexity and the variance of the CTU luminance samples to explore the trade of between energy reduction and compression quality. Experimental results show that the proposed algorithm is able to reduce the energy consumption of the HEVC encoder by 45% to 62% — including the algorithm overhead — for a bit rate increase of between 0.49% and 3.4% respectively. Future work will use this energy reduction technique 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 Min, Cheung RCC (2015) A Fast CU Size Decision Algorithm for the HEVC Intra Encoder. TCSVT 25(5):892–896, DOI 10.1109/TCSVT.2014.2363739
2. Bossen F (2013) Common HM test conditions and software reference configurations
3. Carroll A, Heiser G (2010) An analysis of power consumption in a smartphone
4. Cassa MB, Naccari M, Pereira F (2012) Fast rate distortion optimization for the emerging HEVC standard. In: Picture Coding Symposium (PCS), 2012, IEEE, pp 493–496
5. Chan TF, Golub GH, LeVeque RJ (1979) Updating Formulae and a Pairwise Algorithm for Variances Computing Sample. In: COMPSTAT, Springer Science & Business Media, p 30
6. 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 pp 20–27
7. Feng L, Dai M, Zhao Cl, Xiong Jy (2016) Fast prediction unit selection method for HEVC intra prediction based on salient regions. Optoelectronics Letters 12(4):316–320, DOI 10.1007/s11801-016-6064-8
8. Hackenberg D, Schone R, Ilsche T, Molka D, Schuchart J, Geyer R (2015) An Energy Efficiency Feature Survey of the Intel Haswell Processor. IEEE, pp 896–904, DOI 10.1109/IPDPSW.2015.70
9. JCT-VC (2016) HEVC reference software. https://hevc.hhi.fraunhofer.de/
10. Karczewicz M, Ye Y, Chong I (2008) Rate distortion optimized quantization. In: VCEG-AH21, Antalya Turkey
11. Khan MUK, Shafique M, Henkel J (2013) An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding. In: ICIP, IEEE, pp 1578–1582
12. Khan MUK, Shafique M, Henkel J (2018) Energy Efficient Embedded Video Processing Systems. Springer International Publishing, Cham, DOI 10.1007/978-3-319-61455-7
13. Koivula A, Viitanen M, Lemmetti A, Vanne J, Hmlinen TD (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), IEEE, pp 1250–1254
14. Koivula A, Viitanen M, Vanne J, Hamalainen TD, Fasnacht L (2015) Parallelization of Kvazaar HEVC intra encoder for multi-core processors. In: Signal Processing Systems (SiPS), 2015 IEEE Workshop on, IEEE, pp 1–6
15. Lan C, Xu J, Sullivan GJ, Wu F (2012) Intra transform skipping. In: JCTVC-I0408, Geneva, CH
16. Mercat A, Arrestier F, Hamidouche W, Pelcat M, Menard D (2017) Energy Reduction Opportunities in an HEVC Real-Time Encoder. In: Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, IEEE
17. Mercat A, Arrestier F, Pelcat M, Hamidouche W, Menard D (2017) Prediction of Quad-Tree Partitioning for Budgeted Energy HEVC Encoding. In: Signal Processing Systems (SiPS), 2017 IEEE Workshop on, IEEE
18. MulticoreWare (2017) x265 HEVC Encoder / H.265 Video Codec. http://x265.org/
19. Peng KK, Chiang JC, Lie WN (2016) Low Complexity Depth Intra Coding Combining Fast Intra Mode and Fast CU Size Decision in 3d-HEVC
20. Penny W, Machado I, Porto M, Agostini L, Zatt B (2016) Pareto-based energy control for the HEVC encoder. In: ICIP, IEEE, pp 814–818
21. Qualcomm (2014) Snapdragon 810 processor product brief
22. Ruiz D, Fernndez-Escribano G, Adzic V, Kalva H, Martnez JL, Cuenca P (2015) Fast CU partitioning algorithm for HEVC intra coding using data mining. Multimedia Tools and Applications DOI 10.1007/s11042-015-3014-6
23. Shen L, Zhang Z, An P (2013) Fast CU size decision and mode decision algorithm for HEVC intra coding. Consumer Electronics, IEEE Transactions on 59(1):207–213
24. Sullivan GJ, Ohm JR, Han WJ, 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, DOI 10.1109/TCSVT.2012.2221191
25. Sze V, Budagavi M, Sullivan GJ (eds) (2014) High Efficiency Video Coding (HEVC). Integrated Circuits and Systems, Springer International Publishing, Cham
26. Tan TK, Weerakkody R, Mrak M, Ramzan N, Baroncini V, Ohm JR, Sullivan GJ (2016) Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance. TCSVT 26(1):76–90, DOI 10.1109/TCSVT.2015.2477916
27. UltraVideoGroup (2017) Kvazaar HEVC Encoder. http://ultravideo.cs.tut.fi/#encoder
28. Vanne J, Viitanen M, Hamalainen TD, 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, DOI 10.1109/TCSVT.2012.2223013
29. Vantrix (2017) F265 Open Source HEVC/H.265 Project. http://vantrix.com/f-265-2/
30. Viitanen M, Koivula A, Lemmetti A, Vanne J, Hamalainen TD (2015) Kvazaar HEVC encoder for efficient intra coding. In: ISCAS, IEEE, pp 1662–

1665
31. Wang X, Xue Y (2016) Fast HEVC intra coding algorithm based on Otsu's method and gradient. In: BMSB, IEEE, pp 1–5
32. 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, DOI 10.1109/TCSVT.2003.815165
33. Wien M (2015) High Efficiency Video Coding. Signals and Communication Technology, Springer Berlin Heidelberg, Berlin, Heidelberg
34. 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 pp 1–1, DOI 10.1109/TCSVT.2015.2461991