Engineering Science and Technology, an International Journal 22 (2019) 706–714

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Full Length Article

Homogeneity-based fast CU partitioning algorithm for HEVC intra coding

Mohamed Maazouz [a,b,*], Noureddine Batel [a], Nejmeddine Bahri [c], Nouri Masmoudi [c]

[a] LSEA Research Laboratory, Faculty of Technology, University of Médéa, Algeria
[b] Dep. of Electronics Engineering, University of Blida 1, Algeria
[c] LETI Research Laboratory, National School of Engineering, Sfax, Tunisia

ARTICLE INFO

Article history:
Received 6 September 2018
Revised 6 December 2018
Accepted 28 December 2018
Available online 8 January 2019

Keywords:
HEVC coding
CU partitioning
Complexity reduction
Intra coding

ABSTRACT

High Efficiency Video Coding (HEVC) is a new video coding standard released as a successor for H.264/AVC. It expected to reduce by 50% the bitrate for the same perceptual quality. One of the major contributors to the higher compression performance of HEVC is the introduction of larger Coding Units (CU) with recursive partitioning mechanisms. This encoding performance is accompanied by a high computational complexity, which makes it very difficult to achieve real-time encoding especially if we aim to implement this encoder on embedded platforms. In this context, this paper suggests a fast CU partitioning algorithm for Intra-only (All Intra) configuration. The proposal aims to early terminate CU partitioning for homogeneous regions in the video frame, or skip some depths for high textured regions. The decision of Split/Non-split is based on homogeneity classification algorithm, which allows us to avoid the test of the all depths in order to determine the best CU size. Experimental results confirm that the proposed approach can reduce up to 41% of encoding time in average for different video classes and can reach up to 58% for high homogenous texture video sequences.

1. Introduction

HEVC is the new video codec developed by the Joint Collaborative Team on Video Coding (JCT-VC) in January 2013 [1]. This new standard aims to improve the encoding performance by saving 50% of bitrate with the same visual quality compared to its predecessor the H264/AVC. To achieve this encoding performance, several new features are added to the encoder structure. Moreover, the HEVC has replaced the macroblock structure used in H.264/AVC with a new processing unit called coding tree unit (CTU), which can be recursively split into more flexible coding units (CU) in quad-tree fashion. As a result, the coding efficiency in the HEVC is much better compared to H264/AVC. This achievement in compression efficiency is to the detriment of computational complexity, which represents the major challenge for HEVC integration in embedded platforms and applications requiring real-time processing.

In the last few years, great effort has been devoted to HEVC optimization for the purpose of speeding up the encoder. Practically, three strategies have been used to reduce the HEVC encoding time:

1. Code optimization methods via *Single Instruction Multiple Data* (SIMD) operations in assembly language [2–5]
2. Algorithmic optimization methods [6–11]
3. Parallel implementation methods [12–17]

HEVC reference software model (HM) optimization through SIMD operations provides encoding time reduction without any loss in coding efficiency; however, the obtained results remain deficient and much further from real time running. In the other side, algorithmic optimization methods offer a significant gain penalized by a loss in compression performance. Considering parallel implementations, the compression efficiency is maintained without any losses; however, the problem lies in the encoder parallelization. The HEVC carries different parallelism techniques including Tiles, Slices and Wavefront Parallel Processing (WPP) but these strategies are not implemented in the HM reference software, which is not a multithreaded code. It is to the programmers to envisage a solution for parallel computation.

In this paper, we explore the possibility of speeding up the HEVC encoder for Intra-only configuration using an early CU termination algorithm. Our proposal is based on CU's texture analysis, where we terminate the CU splitting before exploring all possible sizes for CU, which allows us to save the encoding time particularly for high-resolution sequences.

* Corresponding author at: LSEA Research Laboratory, Faculty of Technology, University of Médéa, Algeria
E-mail address: maazouz.mohamed@univ-medea.dz (M. Maazouz).
Peer review under responsibility of Karabuk University.

https://doi.org/10.1016/j.jestch.2018.12.016
2215-0986/

The remainder of the paper is organized as follow: Section 2 gives an overview of HEVC video coding. Section 3 explores some related works according to CU partitioning optimizations. The proposed fast CU size decision is detailed in Section 4. Experimental results of the proposed approach are presented in Section 5. Section 6 summarizes the results of this work and draws conclusions.

2. HEVC overview

2.1. HEVC encoder description

The HEVC coding standard maintains almost the same coding structure as his predecessor H264/AVC. However, HEVC replaces the macro-block structure with a new processing unit called coding tree unit (CTU). Fig. 1 presents a simplified block diagram of the HEVC reference encoder model (HM) [18].

In the encoding side, each frame is split into multiple CTUs. A CTU is a square form with a size from 8 × 8 to 64 × 64, and can be divided into blocks called coding blocks (CU) using a quad-tree algorithm as presented in Fig. 2. Therefore, the size of a CU block can be 64 × 64, 32 × 32, 16 × 16 or 8 × 8. For each CU, two prediction types are performed in order to conclude the best prediction unit (PU) between intra and inter mode according to the (RD_{cost}). The prediction mode, which gives the minimum RD_{cost}, will be selected.

$$RD_{cost} = \lambda_{pred} * R + D \quad (1)$$

Eq. (1) expresses the rate distortion cost in function of the lambda parameter $\lambda pred$ (Lagrange constant given in the HEVC test model) [18], the distortion D, and R the required bitrate to code the CU at the current depth.

For prediction with intra mode, a block is predicted through neighboring pixels using the spatial correlation property between the neighboring blocks. HEVC presents 35 intra prediction modes (DC, planar and 33 directional modes) for the purpose of improving the efficiency of intra prediction as compared to H.264/AVC, which uses only nine intra prediction modes. The intra PU size is a square-shape of 2N × 2N or N × N.

In addition to spatial correlation between neighboring blocks in the same frame, neighbor pictures are also characterized by a high similarity. This temporal correlation propriety between successive frames is exploited to encode the current picture and this part is called inter-prediction. In fact, inter prediction module consists of two parts: Motion Estimation (ME) which aims to determine the best Motion Vector (MV) of the current CU compared to its position in reference frames and then the Motion Compensation (MC) of the best PU in correspondence with the calculated motion vector. The PU for the inter prediction supports square shapes (2N × 2N or N × N) and non-square shapes such as 2N × N, N × 2N, 2N × nU, 2N × nD, nR × 2N, and nL × 2N [19].

Fig. 3 illustrates an example for sub-division of a 64 × 64 CTU into CU, PU, and TU.

If the motion vector difference and the residual block, which is the difference between the current CU and the best prediction block (PU), are equal to 0, the CU is coded in Skip Mode (only for 2N × 2N size). The residual block is fragmented into multiple transform units (TUs) recursively to create a residual quad-tree (RQT). TUs are sub-partitions of a coding unit and can have one of these 32 × 32, 16 × 16, 8 × 8, and 4 × 4 sizes. The next step consists of applying a discrete cosine transform (DCT) and quantification for the residual error to reduce the representative data of the current CU.

Entropy coding represents the next step after DCT transform and quantification. HEVC uses context adaptive binary arithmetic coding (CABAC), the same used in H.264/AVC but with some improvements. A decoding process is also integrated in the HEVC encoder structure. This decoder is based on inverse quantification and inverse transform in order to reconstruct the encoded frame, which will be used afterward as reference frame for the next frames.

Before storing the reconstructed samples in the decoded picture buffer, HEVC needs two further processing steps, i.e. de-blocking filtering (DBF) and sample adaptive offset (SAO) filtering. This post-processing is introduced to reduce the blocking artifacts due to block based coding and improve the quality of the reconstructed pictures.

2.2. Mode decision complexity

The coding unit is characterized by its size and depth. The depth choice in each CTU goes through a decision process based on the RD_{cost} calculation of each CU partition inside the CTU. The recursive

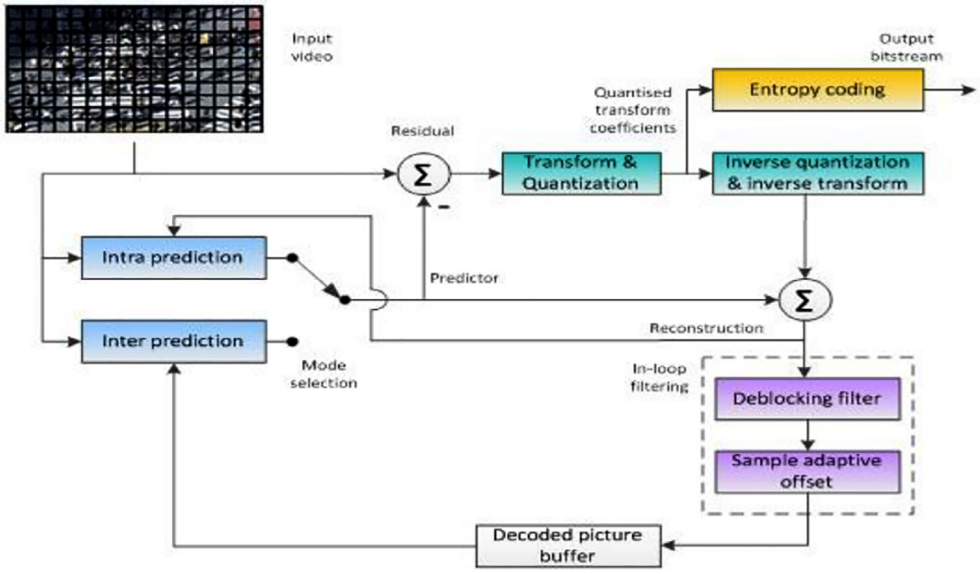

Fig. 1. HEVC block diagram.

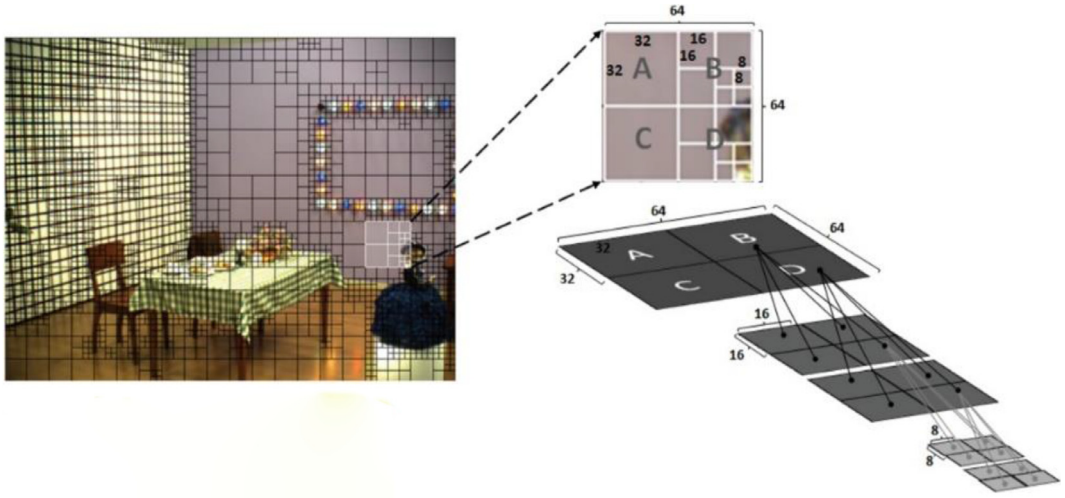

Fig. 2. HEVC hierarchical quad-tree structure.

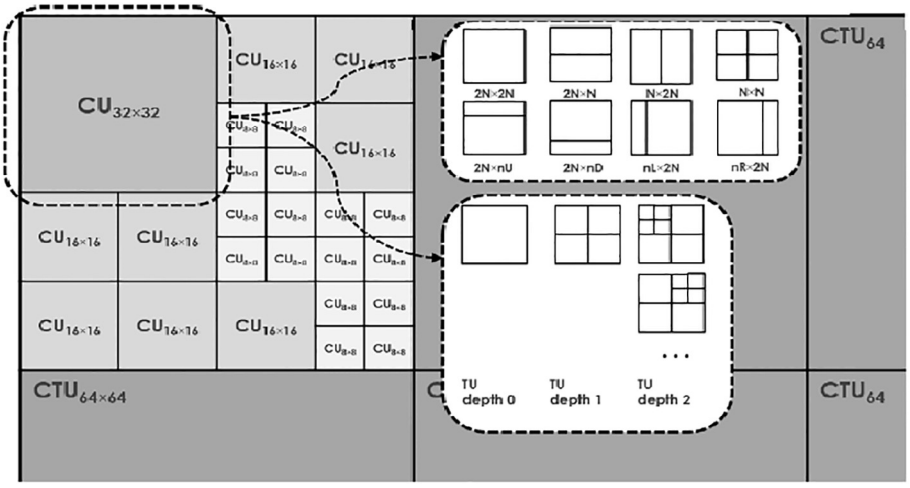

Fig. 3. Example of partitions CTU, CU, PU and TU.

structure is represented by means of a flag sequences called "split flag" which determine the partitioning decision of the coding unit.

In the largest CTU with 64×64 size, called also LCU (Largest Coding Unit), the RD_{cost} RD_1 is calculated since the split flag is set to 0. Then the split flag changes to be 1 and the LCU is partitioned into four sub-partitions CUs of 32×32 size. The first one, CU_10, as presented in Fig. 4, has an RD_{cost} equal to RD_2. Then, we move to the next depth where the CU is partitioned into four CUs with 16×16 of size. RD_3 is the RD_{cost} of the first CU of size 16×16 (CU_20). If its split flag is 1, the last depth (depth = 3) is reached and the CU is therefore partitioned into four Smallest Coding Unit (SCU) of size 8×8. The RD_{cost} for each SCU will be noted RD_4, RD_5, RD_6 and RD_7, respectively. The first decision will be taken from the bottom to the top, determining if the first CU of size 16×16 is chosen or not. We need a comparison of the sum of the four RD_{cost} of the SCU 8x8 with the RD_3 of the CU 16×16 to make a decision. If the RD_3 is greater than the sum of RD_4, RD_5, RD_6 and RD_7, the partitioning decision of CU_20 will be taken, otherwise CU_20 will not be split. Alike for the other CUs, the decision is always based on this equation. Therefore, if the inequality (2) is verified, no split is done.

$$RD_{cost\_CU} < \sum_{k=0}^{3} RD_{cost\_subCU(k)} \quad (2)$$

3. Related works

Quite recently, several works have been worked on the CU partitioning optimizations for HEVC encoders. Partitioning structure in the HEVC standard has a direct impact on encoding performance. In fact, thanks to the different dimensions of CU, PU and TU, the HEVC standard succeeds a better quality/bitrate ratio. However, the encoding time rises according to the number of possible forms of CU, the reason why researchers and developers focus on this step in order to optimize the HEVC encoder.

Many researchers have proposed various algorithms for early CU size decision. Authors in [7] proposed an interesting methodology for accelerating the CU splitting in HEVC intra coding using data mining for off-line classifier training. Their algorithm uses the mean and the variance of CUs and sub-CUs as attributes for the split/non-split decision. This algorithm can decrease the encoding time by over 52% compared to the HM16.6, while the bitrate increase is under 2%.

Another solution is described in [10] that aims to early terminate the CU splitting process based on the Bayesian decision rule using joint online and offline learning. A two class problem was considered to classify CUs into non-partitioning and partitioning classes. This algorithm supports all encoding configurations including Random Access, Low delay and Intra-only.

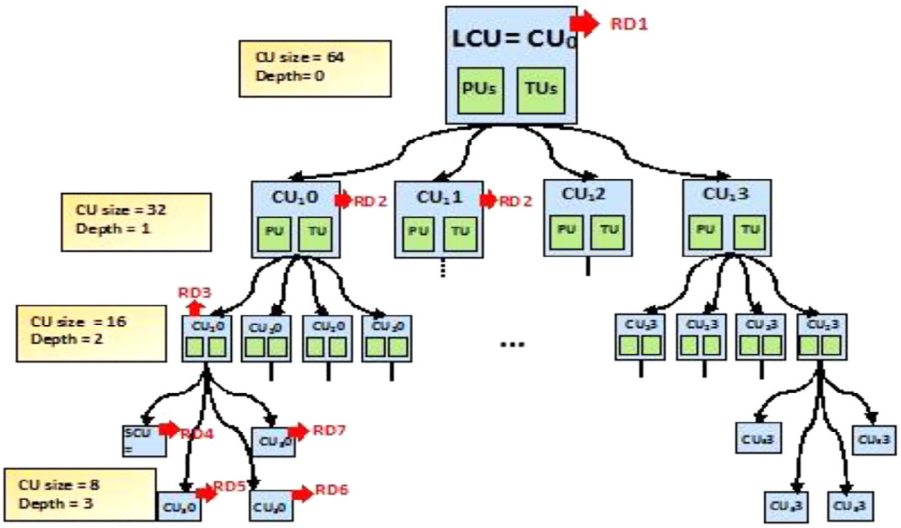

Fig. 4. Structural example of the HEVC quad-tree concept with CU = 64 × 64 and depth = 3.

Shang et al. [9] proposed a fast algorithm for intra prediction using the correlation between both neighboring coding units and prediction units, where the decision split/non-split is made in accordance with the CU texture and neighbors CU size. A fast prediction mode choice is concluded using the correlation between the higher layer mode and neighboring PU modes. A statistical analysis based on the variance operator was adopted by [20] in order to build an off-line threshold data-base used to make split/non-split decision which allows to skip less probable modes. The proposed algorithm was dedicated to Ultra High Definition videos.

In a recent paper presented in [21], HEVC complexity for intra coding is reduced through a fast intra-mode decision and CU size decision. To avoid searching among all 35 prediction modes, the algorithm proposed by authors in [21] is based on the use of the average gradients in the horizontal and vertical directions (AGH, AGV) in order to limit the candidate modes. Besides, CU early split/termination relies on some features such as the difference of depths and the HADAMARD cost ratio between current CU and its adjacent CUs.

Authors in [22] have proposed an algorithm for complexity reduction of HEVC intra prediction using fast mode decision based on the distribution of the dominant edge assent. The decision on the dominant edge is made in accordance to the minimum DEA considering the four direction degree 0, 45, 90, and 135.

The proposed algorithm in [23] aims to reduce the intra coding complexity based on the edge information. A preprocessing stage is introduced to estimate edge magnitude and edge direction for further use in PU partitioning decision. The edge information are assessed using two Sobel convolution kernels (of 3x3 pixels in size). Within a homogenous region, the PU size is likely selected as bigger. Besides, the directional prediction modes are excluded and only DC prediction mode and Planar prediction mode are retained to deduce the optimal prediction mode. Conversely, a complex region is predicted with smaller PU size.

Authors in [24] has also proposed a scheme for fast CU partitioning using Sobel operator in intra coding. First, the gradient vector magnitude is calculated for all pixels in the current LCU, and then a choice is made, based on the thresholded values of the gradient vector magnitude, to decide whether the LCU comprises edges or not. If there is no edge within the LCU, the block is labeled as a smooth area, and subsequently an early CU termination is processed.

In [25], the correlation between a local edge and intra prediction mode is exploited to reduce intra coding complexity. Sobel edge operator is used to detect the four main edge directions (vertical, horizontal, 45° and 135°), this information is used to preselect the prediction mode among the 33 directional modes. Moreover, the authors proposed to merge sub-partitions with the same direction edge to avoid redundant RDO processes.

Several authors [8,26,27] have proposed a variety of algorithms aimed at speeding up the HEVC inter prediction. Their methods benefit from the spatial and/or temporal correlation property between successive images in a video sequence.

4. Proposed fast CU size decision algorithm for HEVC intra-only coding

The introduction of flexible partitioning in the HEVC aims to cover the video content diversity. Fig. 5 depicts an example of CU splitting in HEVC. From this figure, we can note that large blocks fit with smooth areas, while small blocks are used much more with textured areas [28]. According to the assumption above, we propose an efficient algorithm without complex operations to check the CU texture complexity. If a CU block is annotated as smooth, an early termination of CU partitioning is performed, otherwise (textured block) the partitioning is pursued thoroughly. The main steps of our method are illustrated Fig. 6.

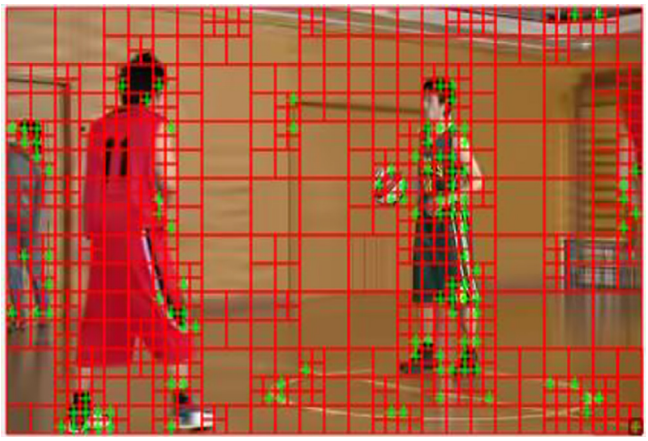

Fig. 5. First frame CU partitioning results of BasketballPass (size: 416 × 240, QP = 37) using HM16.6.

To compress a CTU, the original reference software uses a recursive call to the function xCompressCU, the first call is started with depth = 0 (i.e., CU size equal to the CTU size). After partitioning on four sub-partitions CU, xCompressCU is recalled for each sub-partition with depth = depth + 1, this partitioning and calling process is repeated up to the maximum depth allowed (or the minimum size of a CU is reached).

In our algorithm, the recursive call is maintained only if there is a need for partitioning. For a given CU, we start by computing the maximum of the absolute differences (denoted MAD in flowchart

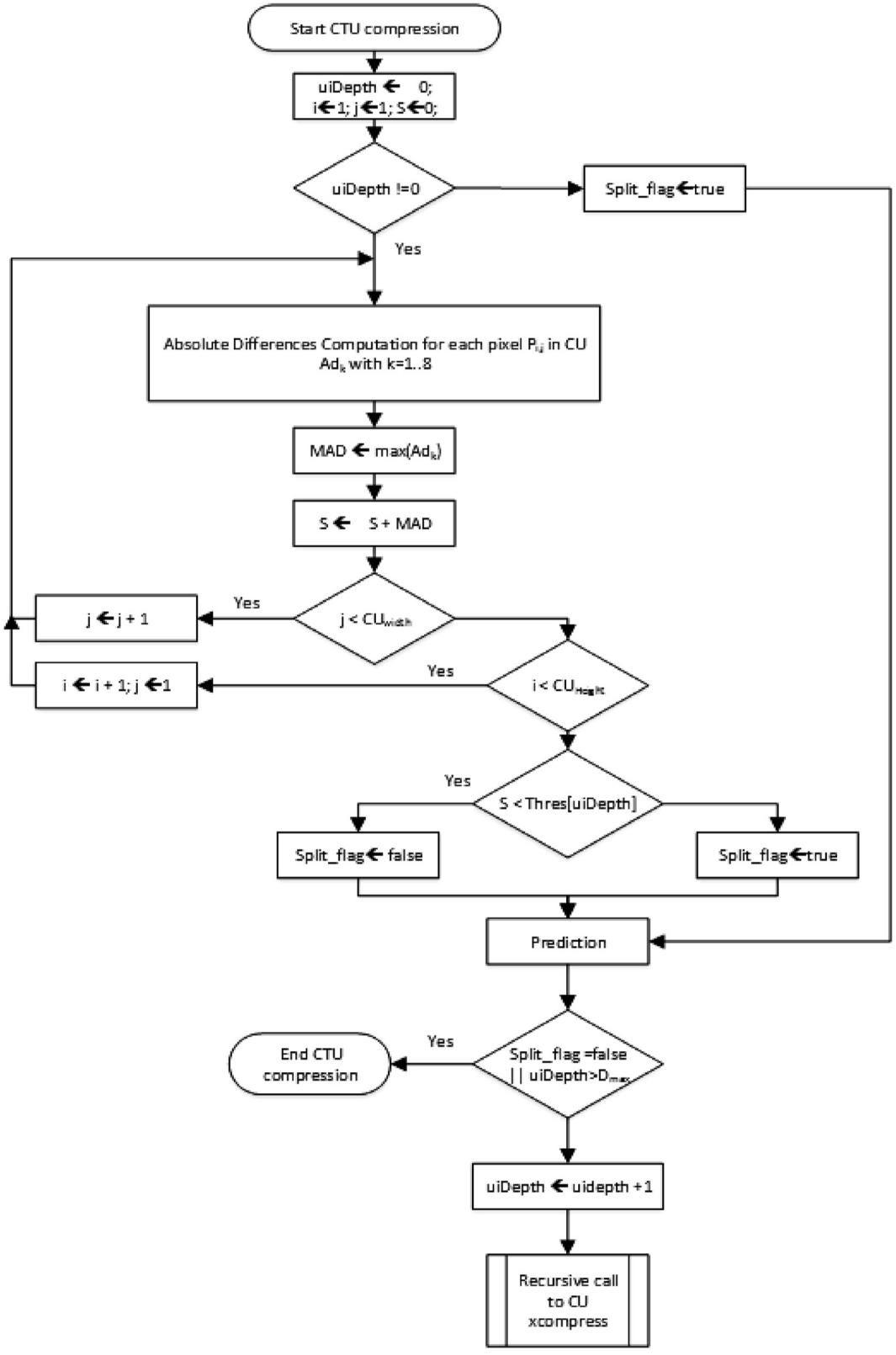

Fig. 6. Proposed fast CU partitioning algorithm.

of Fig. 6) between each pixel $P_{i,j}$ (i and j are the coordinates of the pixel P) and his eight neighbors (pixels highlighted in Fig. 7).

The MAD value is computed as:

$$MAD = \max_{k}(Ad_k), \quad k = 1, 2, \ldots, 8 \tag{3}$$

where:

$$Ad1 = |P_{i,j} - P_{i-1,j-1}|; \quad Ad2 = |P_{i,j} - P_{i-1,j}|$$

$$Ad3 = |P_{i,j} - P_{i-1,j+1}|; \quad Ad4 = |P_{i,j} - P_{i,j-1}|$$

$$Ad5 = |P_{i,j} - P_{i,j+1}|; \quad Ad6 = |P_{i,j} - P_{i+1,j-1}|$$

$$Ad7 = |P_{i,j} - P_{i+1,j}|; \quad Ad8 = |P_{i,j} - P_{i+1,j+1}|$$

Then we calculate the sum (S) of all MADs to make a decision on the homogeneity of the CU at the current depth. If the sum S is greater than a threshold Thres[uiDepth], then the corresponding CU block is annotated as textured and thus, the partitioning can be proceed to the next depth. On the contrary, if the sum S is less than the threshold, the partitioning is terminated. The incomplete CTUs (that does not fit a full LCU) located at the picture boundaries are excluded from the analysis mentioned above because they do not follow the same recursive partitioning scheme.

A preliminary study for the selection of threshold values for each depth was performed in order to ensure a best quality/compression efficiency trade-off.

To show the impact of the proposed approach, we illustrated the results of the application of the homogeneity algorithm on an image in comparison to other algorithms of edge detection, in particular Sobel, Canny and Prewitt algorithms. Fig. 8 shows that the proposed algorithm (with thresholded MAD values) for image texture analysis leads to similar results with thicker edges compared to Sobel and Prewitt filters results.

The threshold determination is very important to achieve such improvements. In our algorithm, the threshold value depends on the size of the considered block (Depth). A set of experiments were conducted in order to pick a threshold value that resolve the trade-off between compression efficiency and encoding time. The presented results in Section 5 are obtained for the following threshold values:

- Depth = 0 (64 × 64): Th1 = 9000
- Depth = 1 (32 × 32): Th2 = 4500
- Depth = 2 (16 × 16): Th3 = 2200

Depth = 3 (8 × 8) is excluded from the homogeneity check because it corresponds to the smallest CU size. We expected a ratio of 1/4 between a threshold at a depth D and a depth D + 1, but experimental tests show that a ratio of ½ can provide better results.

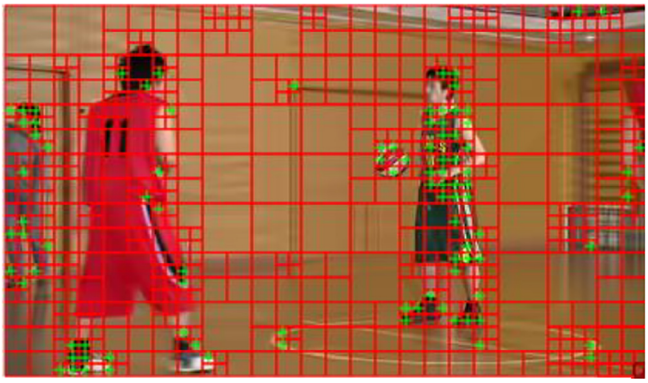

Fig. 7. First frame CU partitioning results of BasketballPass (size: 416 × 240, QP = 37) using the proposed fast CU partitioning algorithm.

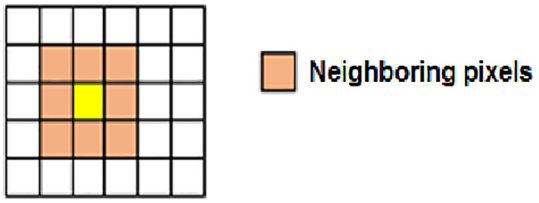

Fig. 8. Considered neighbor pixels for MAD calculation.

5. Experimental results

5.1. Experimental conditions

In this section, the different implementation results are shown. We start with evaluating the performance of our fast CU-size decision algorithm, which constitutes a single-threaded algorithmic optimization. For all our experiments, we used HEVC reference software HM16.6 Main profile under All-Intra configurations, and a subset of sequences recommended by JCT-VC comprising Classes A, B, C, D, and E.

5.2. Evaluation criteria

In our experiments, compression efficiency evaluation is based on the Bjøntegaard Delta [29] rate metric BD-BR (%) and BD-PSNR (dB) together with:

- ΔBR to express the increase in bitrate
- ΔPSNR to express the loss in PSNR

where:

$$\Delta BR\,(\%) = \left(\frac{BR_{proposed}}{BR_{HM16.6}} - 1\right) \times 100 \tag{4}$$

$$\Delta PSNR\,(dB) = PSNR_{proposed} - PSNR_{HM16.6} \tag{5}$$

With $QP \in \{22, 27, 32, 37\}$

The encoding time saving ΔTS for fast CU splitting is defined as:

$$\Delta TS = \left(\frac{EncTime_{proposed}}{EncTime_{HM16.6}} - 1\right) \times 100 \tag{6}$$

5.3. Fast CU size decision results

We started with the implementation of HM16.6 as a reference to evaluate the performance of the proposed method in this work. As can be seen from Fig. 9, the proposed algorithm maintains almost the same partitioning scheme as HM16.6 except for some CUs that are classified as homogeneous, therefore, the CU partitioning is early terminated.

Table 1 summarizes the experimental results of our fast CU size decision algorithm compared to the original HM16.6. Experimental results prove that the proposed fast CU partitioning algorithm allows saving 41% of encoding time in average with a little increase in bitrate by 0.69% as ΔBR and a non-significant quality degradation by 0.05 dB in terms of PSNR. The performance evaluation of the proposed algorithm basically depend on the intrinsic content of the video sequence, and vary from a sequence to another and from a class to another.

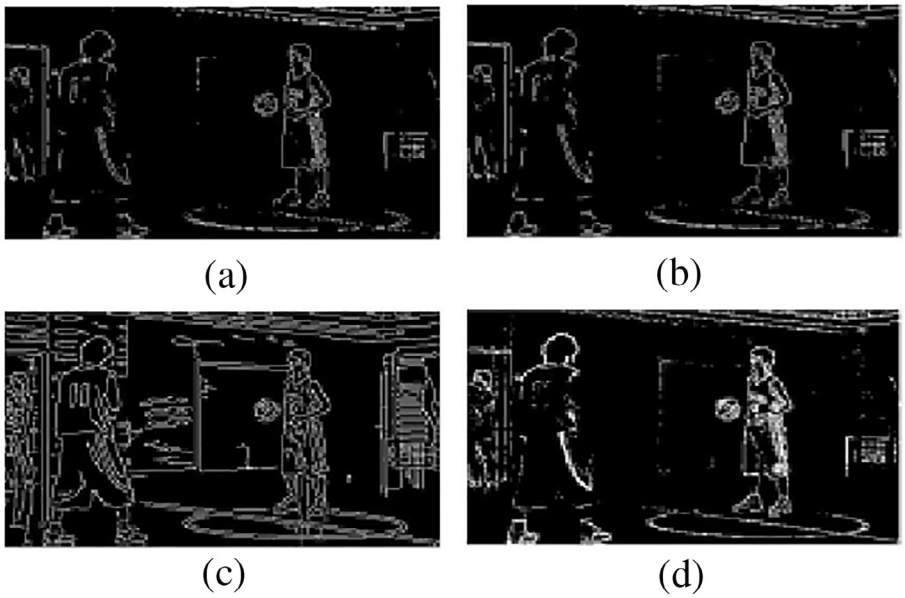

Fig. 9. Results of the edge detection algorithms applied on the first image of the BasketballPass sequence (416 × 240). (a) Sobel filter, (b) Prewitt filter, (c) Canny filter and (d) Homogeneity-based edge detection.

Table 1
Performance evaluation of the proposed fast CU partitioning algorithm compared to HM16.6.

Classes Sequences		BD-BR (%)	BD-PSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔT (%)
Class A	Traffic	1.85	−0.10	0.65	−0.06	−47.50
2560 × 1600	PeopleOnStreet	1.67	−0.10	0.87	−0.05	−43.91
Average A		**1.76**	**−0.10**	**0.76**	**−0.05**	**−45.70**
Class B	Kimono1	1.17	−0.04	0.46	−0.02	−72.95
1920 × 1080	BQTerrace	0.59	−0.03	0.04	−0.03	−31.15
	Cactus	1.10	−0.04	0.03	−0.03	−26.81
	ParkScene	1.07	−0.04	0.04	−0.04	−27.52
	BasketballDrive	1.37	−0.04	−0.15	−0.03	−53.85
Average B		**1.06**	**−0.04**	**0.08**	**−0.03**	**−42.46**
Class E	FourPeople	3.09	−0.18	1.62	−0.08	−51.66
1280 × 720	KristenAndSara	3.25	−0.16	1.78	−0.07	−63.28
	Johnny	3.47	−0.14	2.01	−0.06	−60.73
Average E		**3.27**	**−0.16**	**1.80**	**−0.07**	**−58.56**
Class C	BasketBallDrill	2.80	−0.20	0.90	−0.07	−44.36
832 × 480	BQMall	1.40	−0.09	0.49	−0.04	−30.98
	RaceHorses	1.20	−0.08	0.45	−0.04	−37.64
	PartyScene	0.31	−0.03	0.09	−0.01	−15.74
Average C		**1.43**	**−0.10**	**0.48**	**−0.04**	**−32.18**
Class D	BasketballPass	1.54	−0.09	0.58	−0.04	−42.97
416 × 240	RaceHorses	1.46	−0.13	0.48	−0.05	−26.49
	BlowingBubbles	0.76	−0.06	0.11	−0.03	−18.52
	BQSquare	1.14	−0.15	0.10	−0.05	−22.90
Average D		**1.22**	**−0.11**	**0.32**	**−0.04**	**−27.72**
Average		**1.75**	**−0.10**	**0.69**	**−0.05**	**−41.32**

The best time-saving results are obtained for class E (1280 × 720) with ΔTS up to −63.28%, but the increase in bitrate is the highest among all classes. In our experiments, the class E contains the sequences FourPeople, KristenAndSara and Johnny which are captured by a fixed camera (fixed background with many homogeneous blocks), and the movement of people, in the scene, is relatively slow which creates a strong temporal correlation. Accordingly, our algorithm classifies most of the CTUs as homogeneous and encodes them at a low depth (i.e., large CU size). In the other side, CTUs belonging to class D video sequences (low resolution: 416 × 240) are generally high textured and our proposed algorithm encodes them at a high depth (CUs of size 8 × 8 and 4 × 4 of PU size, hence the computational complexity increases) which decreases the time saving.

In general, our proposal is much more suitable for high-resolution sequences, where a CTU block covers a small region of the image and therefore a more homogeneous content. The performance comparison of the proposed approach with some state-of-the-art algorithms is given in Table 2. As clear, our method outperforms the algorithms proposed in [24] in terms of both compression efficiency and time reduction.

In term of time saving, our algorithm ensures superior results compared to [9,30–32]. However, results in [11] show a time saving about 53% which is better than our time reduction but the

Table 2
Performance comparison of the proposed fast CU partitioning with some related works.

Methods	Average BD-BR	Average BD-PSNR	Average ΔBR	Average ΔPSNR	Average ΔT (%)
[24]	–	–	0.76	−0.09	−31.30
[30]	0.83	–	–	–	24.00
[9]	0.66	−0.04	–	–	−37.91
[31]	0.01	–	–	–	−14.40
[11]	–	–	1.98	−0.13	−53.52
[32]	0.51	–	–	–	−28.12
Proposed	**1.75**	**−0.10**	**0.69**	**−0.05**	**−41.32**

increase in bitrate ΔBR of 1.98% in average is about three times much higher than our result (ΔBR = 0.69%). For high definition sequences, such as class A (1560×1600) and class B (1920×1080), the proposed algorithm for fast CU partitioning decreases the encoding time efficiently with tolerable increase in bitrate.

6. Conclusion

This paper presents an efficient homogeneity-based algorithm for complexity reduction of HEVC intra coding. It has been found that large size blocks are more suitable for encoding homogeneous regions; while textured areas are often encoded by small blocks. To anticipate the partitioning of the CTU blocks, our algorithm starts with the analysis of the CU blocks for the current depth in order to check the texture complexity. If the block is labeled homogeneous then we stop the partitioning process, otherwise, the partitioning continues in an ordinary way as for the reference software. Summing up the results, it can be concluded that the proposed algorithm for fast CU partitioning achieves in average 41% encoding time saving with average 1.75% and 0.69% increment in BD-BR and ΔBR, respectively, and negligible loss in PSNR. Compared to some recent state-of-the-art algorithms, our algorithm gives better encoding time saving or less bitrate losses.

As a perspective for the presented work, the results can be improved by adopting an algorithm for the preliminary estimation of the depth based on a statistical analysis. Another improvement can be very interesting dealing with the estimation of the prediction mode and avoiding search in all possible prediction directions. Finally, performance evaluation for low-delay (LD) and random access (RA) configurations will be more desirable.

References

[1] V. Sze, M. Budagavi, G.J.S. Editors, High Eciency Video Coding (HEVC): Algorithms and Architectures, 2014. doi:10.1007/978-3-319-06895-4.

[2] K. Shah, Final Report Time Optimization of HEVC Encoder over X86 Processors using SIMD Spring 2013 Multimedia Processing EE5359 Advisor : Dr. K.R. Rao Department of Electrical Engineering University of Texas, Arlington Kushal Shah, 2013.

[3] Y.J. Ahn, T.J. Hwang, D.G. Sim, W.J. Han, Implementation of fast HEVC encoder based on SIMD and data-level parallelism, Eurasip J. Image Video Process. 2014 (2014) 1–19, https://doi.org/10.1186/1687-5281-2014-16.

[4] K. Chen, Y. Duan, L. Yan, J. Sun, Z. Guo, Efficient SIMD optimization of HEVC encoder over X86 processors, Apsipa ASC (2012) 1–4.

[5] A. Lemmetti, A. Koivula, M. Viitanen, J. Vanne, T.D. Hämäläinen, AVX2-optimized Kvazaar HEVC intra encoder, in: Proc. – Int. Conf. Image Process. ICIP, 2016, pp. 549–553, https://doi.org/10.1109/ICIP.2016.7532417.

[6] Y. Kim, D. Jun, S. Jung, J.S. Choi, J. Kim, A fast intra-prediction method in HEVC using rate-distortion estimation based on hadamard transform, ETRI J. 35 (2013) 270–280, https://doi.org/10.4218/etrij.13.0112.0223.

[7] D. Ruiz, G. Fernández-Escribano, V. Adzic, H. Kalva, J.L. Martínez, P. Cuenca, Fast CU partitioning algorithm for HEVC intra coding using data mining, Multimed. Tools Appl. 76 (2017) 861–894, https://doi.org/10.1007/s11042-015-3014-6.

[8] K. Goswami, B.-G. Kim, D. Jun, S.-H. Jung, J.S. Choi, Early coding unit-splitting termination algorithm for high efficiency video coding (HEVC), ETRI J. 36 (2014) 407–417, https://doi.org/10.4218/etrij.14.0113.0458.

[9] X. Shang, G. Wang, T. Fan, Y. Li, CU Fast size decision and PU mode decision algorithm in HEVC intra coding, IEEE Int. Conf. Image Process IEEE 2015 (2015) 1593–1597, https://doi.org/10.1109/ICIP.2015.7351069.

[10] H.-S. Kim, R.-H. Park, Fast CU partitioning algorithm for HEVC using an online-learning-based bayesian decision rule, IEEE Trans. Circuits Syst. Video Technol. 26 (2016) 130–138, https://doi.org/10.1109/TCSVT.2015.2444672.

[11] K. Lim, J. Lee, S. Kim, S. Lee, Fast PU skip and split termination algorithm for HEVC intra prediction, IEEE Trans. Circuits Syst. Video Technol. 25 (2015) 1335–1346, https://doi.org/10.1109/TCSVT.2014.2380194.

[12] P. Piñol, H. Migallón, O. López-Granado, M.P. Malumbres, Parallel strategies analysis over the HEVC encoder, J. Supercomput. 70 (2014) 671–683, https://doi.org/10.1007/s11227-014-1121-1.

[13] A. Koivula, M. Viitanen, J. Vanne, T.D. Hamalainen, L. Fasnacht, Parallelization of Kvazaar HEVC intra encoder for multi-core processors, IEEE Work. Signal Process. Syst IEEE 2015 (2015) 1–6, https://doi.org/10.1109/SiPS.2015.7345015.

[14] C. Yan, Y. Zhang, J. Xu, F. Dai, J. Zhang, Q. Dai, F. Wu, Efficient parallel framework for HEVC motion estimation on many-core processors, IEEE Trans. Circuits Syst. Video Technol. 24 (2014) 2077–2089, https://doi.org/10.1109/TCSVT.2014.2335852.

[15] M. Maazouz, N. Bahri, N. Batel, A. Toubal, N. Masmoudi, Parallel implementation of Kvazaar HEVC on multicore ARM processor, in: 2016 8th Int. Conf. Model. Identif Control, IEEE, 2016, pp. 1086–1091, https://doi.org/10.1109/ICMIC.2016.7804275.

[16] Chenggang Yan, Yongdong Zhang, Feng Dai, Liang Li, Highly parallel framework for HEVC motion estimation on many-core platform, in: 2013 Data Compression Conf., IEEE, 2013, pp. 63–72, https://doi.org/10.1109/DCC.2013.14.

[17] F. Amish, E.B. Bourennane, Fully pipelined real time hardware solution for High Efficiency Video Coding (HEVC) intra prediction, J. Syst. Archit. 64 (2016) 133–147, https://doi.org/10.1016/j.sysarc.2015.10.002.

[18] Il-Koo Kim, High efficiency video coding (HEVC) test model 15 (HM15) encoder description, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 17th Meeting: Valencia, ES, 27 Mar. – 4 Apr, 2014, 2014.

[19] I.K. Kim, J. Min, T. Lee, W.J. Han, J.H. Park, Block partitioning structure in the HEVC standard, IEEE Trans. Circuits Syst. Video Technol. 22 (2012) 1697–1706, https://doi.org/10.1109/TCSVT.2012.2223011.

[20] N. Dhollande X. Ducloux O. Le Meur C. Guillemot Fast block partitioning method in HEVC intra coding for UHD video 5th IEEE Int. Conf. Consum. Electron. – Berlin, ICCE-Berlin 2015 2016 10 11 10.1109/ICCE-Berlin.2015.7391205

[21] T. Zhang, M.T. Sun, D. Zhao, W. Gao, Fast intra-mode and CU size decision for HEVC, IEEE Trans. Circuits Syst. Video Technol. 27 (2017) 1714–1726, https://doi.org/10.1109/TCSVT.2016.2556518.

[22] Y. Yao, X. Li, Y. Lu, Fast intra mode decision algorithm for HEVC based on dominant edge assent distribution, Multimed. Tools Appl. 75 (2016) 1963–1981, https://doi.org/10.1007/s11042-014-2382-7.

[23] W. Shi, X. Jiang, T. Song, T. Shimamoto, Edge information based fast selection algorithm for intra prediction of HEVC, in: IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Ishigaki, 2014, pp. 17–20, https://doi.org/10.1109/APCCAS.2014.7032708.

[24] F. Belghith, H. Kibeya, M.A. Ben Ayed, N. Masmoudi, Fast coding unit partitioning method based on edge detection for HEVC intra-coding, Signal Image Video Process. 10 (2016) 811–818, https://doi.org/10.1007/s11760-015-0820-2.

[25] S. Na, W. Lee, K. Yoo, Edge-based fast mode decision algorithm for intra prediction in HEVC, in: 2014 IEEE Int. Conf. Consum. Electron, IEEE, 2014, pp. 11–14, https://doi.org/10.1109/ICCE.2014.6775887.

[26] M. Asfandyar, M. Nawaz, M. Hussain, CU Accelerated decision based on enlarged CU sizes for HEVC UHD videos, in: 2016 IEEE Int. Conf. Signal Image Process, IEEE, 2016, pp. 374–378, https://doi.org/10.1109/SIPROCESS.2016.7888287.

[27] S. Ahn, B. Lee, M. Kim, A novel fast CU encoding scheme based on spatiotemporal encoding parameters for HEVC inter coding, IEEE Trans. Circuits Syst. Video Technol. 25 (2015) 422–435, https://doi.org/10.1109/TCSVT.2014.2360031.

[28] M.T. Pourazad, C. Doutre, M. Azimi, P. Nasiopoulos, HEVC: the new gold standard for video compression: how does HEVC compare with H.264/AVC?,

IEEE Consum Electron. Mag. 1 (2012) 36–46, https://doi.org/10.1109/MCE.2012.2192754.
[29] G. Bjontegaard, Calculation of average PSNR differences between RD-curves, Document VCEG-M33, 13th Meeting: Austin, Texas, USA, 2–4 April, 2001.
[30] Jongho Kim, Yoonsik Choe, Yong-Goo Kim, Fast coding unit size decision algorithm for intra coding in HEVC, in: 2013 IEEE Int. Conf. Consum. Electron, IEEE, 2013, pp. 637–638, https://doi.org/10.1109/ICCE.2013.6487050.
[31] M. Ramezanpour Fini, F. Zargari, Two stage fast mode decision algorithm for intra prediction in HEVC, Multimed. Tools Appl. 75 (2016) 7541–7558, https://doi.org/10.1007/s11042-015-2675-5.
[32] D.G. Fernandez, A.A. Del Barrio, G. Botella, C. Garcia, Fast CU size decision based on temporal homogeneity detection, in: 2016 Conf. Des. Circuits Integr. Syst., 2016, pp. 1–6, https://doi.org/10.1109/DCIS.2016.7845379.