TITAN: Tile Timing-Aware Balancing Algorithm for Speeding Up the 3D-HEVC Intra Coding

Mário Saldanha[1], Gustavo Sanchez[2,3], Bruno Zatt[1], César Marcon[2], Luciano Agostini[1]
[1]Federal University of Pelotas – Pelotas, Brazil
[2]Pontificial Catholic University of Rio Grande do Sul – Porto Alegre, Brazil
[3]IF Farroupilha – Alegrete, Brazil
{mrdfsaldanha, zatt, agostini}@inf.ufpel.edu.br, gustavo.sanchez@acad.pucrs.br, cesar.marcon@pucrs.br

***Abstract*—This paper presents the Tile Timing-Aware balancing algorithm (TITAN) for speeding up the 3D-High Efficiency Video Coding (3D-HEVC) video encoding. The design of the TITAN algorithm is based on the premise that the encoding time of tiles partitioning of neighbor frames are similar. Therefore, based on the encoding time of the last-encoded frame, TITAN controls the tiles boundaries aiming to maximize the balance among tiles. Our software evaluation with TITAN implemented in 3D-HEVC Test Model 16.0 achieved an average of 6.2% higher speedup compared to the uniform-sized tiles. This is the first work in the literature proposing to speed up the 3D-HEVC video encoding by balancing the tiles workload.**

Keywords— 3D-HEVC, parallelism, tiles, multicore systems.

I. INTRODUCTION

Significant advances in video coding were obtained with the standardization of the High Efficiency Video Coding (HEVC) [1], making it possible to encode efficiently high-resolution videos using few bits while maintaining the video quality. With the increase of interest in 3D video applications, these advances were also extended for 3D video coding employing encoders such as the 3D-High Efficiency Video Coding (3D-HEVC) [2].

To deal with 3D video coding challenges, 3D-HEVC uses the HEVC functionalities and adopts the Multiview Video plus Depth (MVD) data format [3]. MVD encompasses texture and depth map information, where each texture frame is associated with a corresponding depth map indicating the distance of objects from the camera. The depth maps are represented by grayscale images and characterized by having large homogeneous regions and sharp edges. The depth maps associated with texture data allows the generation of a dense set of synthesized views using lightweight techniques such as Depth Image Based-Rendering (DIBR) [4]. Thus, 3D-HEVC can achieve higher compression rate with high video quality since no intermediary views are needed to be encoded and transmitted.

However, the 3D-HEVC encoder deals with a high complexity to encode different views simultaneously and the high computational effort to perform effective predictions for depth maps encoding. Because of its high computational cost, 3D-HEVC also inherits structures from HEVC to facilitate parallel processing. These structures allow different levels of parallelization and are called Slices, Wavefront Parallel Processing (WPP) and Tiles [5]. These parallelization mechanisms were developed because HEVC is not able to reach a high processing rate in a single core execution even using complexity reduction solutions such as [6] or dividing the HEVC quadtree evaluation using a multicore processor, where a speedup of only 16 was obtained with a 64-core processor [7].

This paper focuses on tiles, which are rectangular picture partitions defined by vertical and horizontal boundaries of Coding Tree Units (CTUs). Each tile is encoded individually, breaking the dependencies among CTUs. Thus, exploring the parallelism of multicore systems to speed up the 3D-HEVC encoding, but increasing the size of the encoded video due to the breakdown dependencies. Besides, the workload of the processors may be unbalanced among the tiles. This behavior happens because complex regions of a picture (e.g., high movement intensity and sudden variation in sample values) tend to demand more processing than simpler regions (e.g., low movement intensity and smooth transitions in sample values). Thus, when distributing the tiles processing into multiple cores the non-balanced distribution results in lower speedup since some cores finish their processing later than other cores.

Some works, such as [8]-[10], focus on parallel video coding using tiles. Blumenberg et al. [8] developed a technique to set the tile partitions dynamically aiming to reduce the encoding efficiency losses caused by the use of tiles. However, this technique considers only encoding efficiency and may leading to unbalanced workload distribution among tiles, limiting the speedup. Storch et al. [9] and Koziri et al. [10] proposed speedup-tiling algorithms based on the previously encoded CTUs. Their proposals use the workload history information to define the vertical and horizontal boundaries of the tiles.

Although these works achieve relevant speedup results, they use the time of each encoded CTU, which tend to be more susceptible to workload variations. Besides, all these solutions were developed focusing on 2D video coding context considering HEVC encoder. To the best of our knowledge, there is no work focusing on tiles for the 3D-HEVC encoder. This paper presents Tile Timing-Aware balancing (TITAN), a novel algorithm for speeding up the 3D-HEVC intra coding. TITAN algorithm aims to control the tile boundaries based on the tiles workload of the previously encoded frame. We selected intra coding because 3D-HEVC has significantly increased its encoding effort compared to its predecessor standard [11].

II. PARALLEL ENCODING BACKGROUND AND ANALYSIS

The use of parallel processing in multicore/manycore systems enables to speed up the 2D/3D video coding. Besides, some features of 3D systems are inherently parallel, such as having several views representing the same content that could

978-1-7281-0397-6/19/$31.00 ©2019 IEEE

be encoded at the same time. Thus, to improve the 3D video processing, 3D-HEVC inherits from HEVC the parallel approaches Slices, Tiles, and WPP.

Slices is a parallel encoding strategy introduced by the well-known H.264/AVC to prevent encoding losses in the case of transmission errors and allow parallelizing the encoded/decoded frames. When the slice strategy is enabled each frame is divided into one or more slices grouping a set of sequential CTUs, where each slice is independently decoded; i.e., without using any information from any other slice. However, headers into bitstream are necessary to identify each slice incurring into significant encoding losses, mainly when using many slices.

WPP technique is also employed as a strategy for 3D-HEVC parallel encoding. The parallelization is applied in each row of a picture without breaking dependencies among the encoded CTUs. Consequently, WPP divides each picture into a row of CTUs and each row is considered as an execution flow. However, when a new CTU is being encoded and depends on information of a CTU from a different execution flow, the encoding flow of that CTU is not performed until dependencies are satisfied. Thus, smaller encoding efficiency losses are noticed at the impact of parallelization inefficiencies.

Tiles approach allows creating picture partitions processed in parallel without incurring significant encoding losses. Such as slices, the tiles boundaries also break the inter-blocks dependencies. Among the parallelism encoding tools, tiles offer the possibility of independent encoding for each structure partitioning and allows exploring several levels of parallelism without incurring in significant encoding efficiency losses. The tile structure can provide different levels of parallelization since it allows the division of the encoding frame into a different number of CTUs to compose each tile and the rectangular partitioning tend to group CTUs highly correlated.

The workload of the tiles tends to be unbalanced during the encoding since different video sequences and frames have different characteristics. Besides, in 3D-HEVC context, texture and depth maps provides different types of information, implying in different workload for the frame captured in the same view at the same time. Fig. 1 presents an analysis of the encoding the first 100 frames in Shark video sequence, with all-intra encoder configuration and using QP 25 for texture and 34 for depth maps, defining 3×3 tiles with uniform spacing. Two premises can be taken from the analyses that were used in the design of the TITAN algorithm.

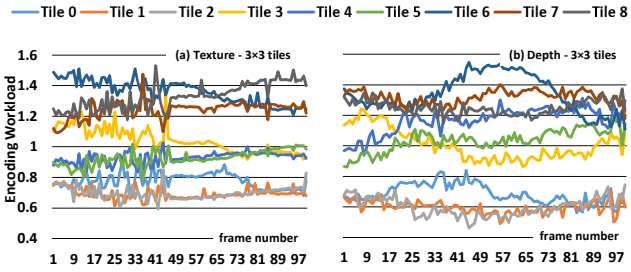

Fig. 1 – Workload analysis for 3×3 uniform tiles encoding Shark video sequence using QP=(25, 34) for (a) texture and (b) depth map.

Premise (i) **-** The diversity of characteristics of each frame in a video implies that each region of a frame requires a specific workload. Thus, associating tiles with frames with uniform size implies in a low speedup when executing in homogeneous cores.

Premise (ii) **-** When encoding the current frame using the same tile size distribution from the previous frame, the encoding workload tends to be similar. Since the current frame workload would be similar, then, a predictive algorithm can consider the last frame workload in its decision for tile balancing.

III. TITAN ALGORITHM

The main idea of the TITAN algorithm is moving the tiles boundaries according to the previous frame workload distribution, since balancing the workload speeds up the frame encoding and the distribution of the previously encoded frame contains significant similarity with the current encoding frame.

Fig. 2 exemplifies 12 uniform tiles mapped into a 4K ultra HD frame (i.e., 3840×2160 pixels or 60×34 CTUs), showing the nomenclature used in the TITAN description and defined next.

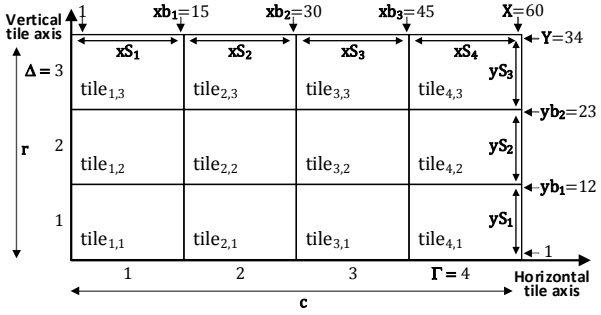

Fig. 2 – An example of 12 uniform tiles mapped into a 4K ultra HD frame.

Let $X \times Y$ be the CTU matrix of a given frame, where X and Y represent the vertical and horizontal dimensions, respectively. Let $\Gamma \times \Delta$ be the tile matrix of the same frame, where Γ and Δ represent the vertical and horizontal dimensions, respectively. Let (c, r) be a pair identifying the column and row of a tile in the frame (i.e., the horizontal and vertical positions), respectively, such that $1 \leq c \leq \Gamma$ and $1 \leq r \leq \Delta$; then, $tile_{c,r}$ is the tile placed on column c, row r. Besides, each tile is a structure encompassing $xS_c \times yS_r$ CTUs, such that $X = \sum_{c=1}^{\Gamma} xS_c$ and $Y = \sum_{r=1}^{\Delta} yS_r$. All tiles can change the horizontal (xS_c) and vertical (yS_r) sizes according to the workload balancing provided by TITAN; but all tiles of a given column c have the same yS_r, and all tiles of a given row r have the same xS_c, performing a regular matrix of tiles. Finally, let xb_c and yb_r be the rightmost horizontal boundary and the upper vertical boundary of a $tile_{c,r}$, respectively. TITAN moves xb_c into the range $1 \leq c < \Gamma$ and yb_r into the range $1 \leq r < \Delta$ to change the quantity of CTUs a column and a row of tiles have, respectively, to balance the tile workload. Fig. 3 shows the TITAN flow chart, which encompasses three main tasks: (i) *Tile Mapping*, (ii) *Vertical Balancing* and (iii) *Horizontal Balancing*.

Tile Mapping is performed once at the beginning of the algorithm. TITAN starts reading X, Y, Γ and Δ, to create a matrix of tiles, each one composed by the same number of CTUs. Since, at the first moment, TITAN has no workload reference, $xS_c = X/\Gamma \ \forall 1 \leq c \leq \Gamma$, and $yS_r = Y/\Delta \ \forall 1 \leq r \leq \Delta$. Note that TITAN regards a symmetric multiprocessing architecture; thus,

any CTU can be mapped to any processor considering the same computation and communication costs.

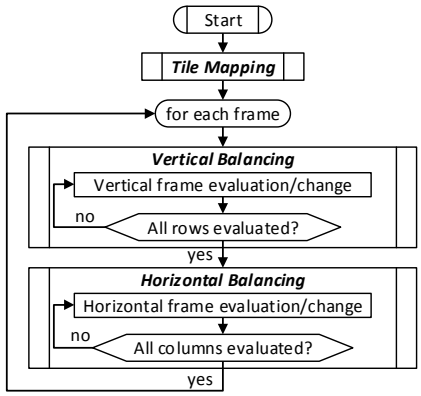

Fig. 3 – TITAN flow chart.

Vertical Balancing and *Horizontal Balancing* are performed at each new frame evaluated by TITAN. The balancing uses (1) to compute the workload of each tile of the frame ($timeTile_{c,r}$), which is performed by summing the computation time of all CTUs inside the $tile_{c,r}$ for encoding the last frame.

$$timeTile_{c,r} = \sum_{j=1}^{xS_c} \sum_{k=1}^{yS_r} timeCTU_{j,k} \quad (1)$$

Vertical Balancing implements a loop from $r = \Delta\text{-}1$ down to 1; each iteration it divides the frame into the set of tiles above and the set of tiles below or equal to yb_r. Equations (2) and (3) compute the sum of the encoding time spent by the set of tiles above ($timeAboveRow_r$) and below or equal to yb_r ($timeBelowRow_r$), respectively, using r as index.

$$timeAboveRow_r = \sum_{\gamma=1}^{\Gamma} \sum_{\delta=r+1}^{\Delta} timeTile_{\gamma,\delta} \quad (2)$$

$$timeBelowRow_r = \sum_{\gamma=1}^{\Gamma} \sum_{\delta=1}^{r} timeTile_{\gamma,\delta} \quad (3)$$

Equation (4) calculates the workload fraction above r (wa_r) spent during the last frame encoding, while (5) computes the target workload desired above r (ta_r) for a perfect balancing. Equation (6) estimates the workload of a row of CTUs above yb_r (wr_r), which is used to avoid that when trying to balance, a larger workload is caused for the other side, worsening the balancing; we use $wr_r/2$ to check if the balancing must be performed.

$$wa_r = \frac{timeAboveRow_r \times 100\%}{timeAboveRow_r + timeBelowRow_r} \quad (4)$$

$$ta_r = \frac{(\Delta - r) \times 100\%}{\Delta} \quad (5)$$

$$wr_r = \frac{timeAboveRow_r \times 100\%}{\Delta - r} \quad (6)$$

If $ta_r < wa_r + wr_r/2$, then increasing yb_r tends to provide a better workload balancing; otherwise, nothing is done. This process is symmetric in the case $ta_r > wa_r + wr_r/2$; however, the evaluations are performed reducing yb_r.

Horizontal Balancing is analogous to *Vertical Balancing*, but dividing the frame vertically and evaluating the set of tiles left and the set of tiles right or equal to xb_c. Note that, although TITAN starts with *Vertical Balancing*, the balancing order does not change the balancing results.

Fig. 4 exemplifies the usage of the TITAN algorithm in a frame divided into three parts, implying $ta_2 = 33.33\%$ and $ta_1 = 66.66\%$. Since $wa_2 = 40\%$ and $wa_1 = 75\%$, then $ta_2 < wa_2$ and $ta_1 < wa_1$ making the algorithm increases yb_2 and yb_1. Each move represents 5% of the encoding time; thus, decreasing and increasing the tiles in rows 3 and 1, respectively.

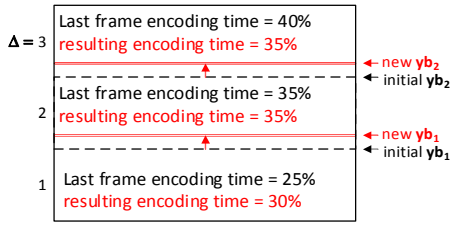

Fig. 4 – Example of the TITAN algorithm for balancing three rows of tiles.

IV. Experimental Evaluation

The TITAN algorithm was implemented into 3D-HEVC Test Model (3D-HTM) 16.0 [12] and evaluated according to the Common Test Conditions (CTC) for 3D experiments [13] using all-intra encoder configuration.

The evaluation was performed using 2×2, 3×3, and 4×4 tiles distributions. Table I shows the Bjontegaard Delta-rate (BD-rate) [14] (synthesized views) obtained using uniform-sized tiles (i.e., all tiles have the same size – without balancing) and the TITAN balancing (i.e., tile size balance according to the TITAN algorithm). On the one hand, the BD-rate of the uniform-sized tiles is equivalent to the TITAN balancing since the variations were small. It happens because TITAN was designed focusing on balancing the workload; therefore, it was already expected that the BD-rate maintains similar to the uniform-sized tiles. On the other hand, considering that the algorithm produces a better balancing, we compared uniform-sized tiles to TITAN balancing regarding the allowed speedup obtained per frame. The average results are presented in Table II. The speedup was computed using (7), where $timeTile_i$ is the time spent to encode the $Tile_i$ and $Highest(timeTile)$ is the highest time spent to encode a tile in the current frame, since the synchronization of the frame depends on the latest tile execution.

The average results show that in a 2×2 distribution, TITAN obtains a speedup of 3.75 in texture, while the uniform-sized tiles obtain 3.65. The gain with TITAN is more expressive with depth maps, obtaining a speedup of 3.68 in TITAN and 3.47 with uniform-sized tiles. The evaluations performed in 3×3 and 4×4 tiles distribution demonstrate that TITAN can scales for more tiles division, obtaining a speedup of 7.71/7.63 (texture/depth) and 13.6/12.5 for the first and the second distribution, respectively, while the uniform-sized tiles obtained 7.42/7.12 and 12.60/11.39 for the same distributions. Once more, the gains obtained with TITAN on depth maps are more expressive than texture compared to the uniform-sized tiles, since it obtains lower depth maps speedup. In some evaluations, the speedup provided by the uniform-sized tiles was higher than the TITAN

balancing. We highlighted these cases in light red color; while the cases TITAN has better results are highlighted with light green color. In most of the cases, TITAN provides a significant speedup.

Table I – BD-rate results for uniform-sized tiles and the TITAN balancing.

Videos	Uniform-sized tiles			TITAN balancing		
	2×2	3×3	4×4	2×2	3×3	4×4
Balloons	1.7%	3.1%	5.4%	1.6%	3.3%	5.4%
Kendo	3.3%	5.8%	8.0%	3.3%	5.8%	8.0%
Newspaper	1.4%	2.7%	5.2%	1.4%	2.8%	5.2%
GT_Fly	1.2%	2.6%	3.9%	1.3%	2.5%	3.9%
PHall2	3.2%	6.1%	9.7%	3.1%	6.3%	9.7%
PStreet	1.2%	2.1%	2.5%	1.1%	2.1%	3.4%
Dancer	0.6%	1.2%	1.7%	0.6%	1.2%	1.7%
Shark	1.7%	1.4%	2.1%	1.7%	1.5%	2.2%
Average	**1.7%**	**3.1%**	**4.8%**	**1.6%**	**3.2%**	**4.9%**

$$Speedup = \frac{\sum_{\gamma=1}^{\Gamma}\sum_{\delta=1}^{\Delta} timeTile_{\gamma,\delta}}{Highest(timeTile)} \quad (7)$$

Table II – Speedup results for 3D-HTM using 2×2, 3×3 and 4×4 tile distributions with uniform-sized tiles and TITAN balancing.

	Videos	Uniform-sized tiles			TITAN balancing		
		2×2	3×3	4×4	2×2	3×3	4×4
Texture	*Balloons*	3.82	7.43	14.20	3.89	7.50	14.47
	Kendo	3.70	7.39	13.86	3.79	7.31	14.20
	Newspaper	3.69	7.18	13.54	3.74	7.72	14.18
	GT_Fly	3.68	7.72	12.57	3.68	7.70	13.04
	PHall2	3.68	7.95	12.53	3.67	7.86	12.86
	PStreet	3.85	7.86	13.04	3.78	7.93	13.41
	Dancer	3.45	6.92	10.70	3.75	7.74	13.47
	Shark	3.35	6.88	10.32	3.74	7.89	13.31
	Average	**3.65**	**7.42**	**12.60**	**3.75**	**7.71**	**13.62**
Depth	*Balloons*	3.44	6.56	10.79	3.64	7.56	11.18
	Kendo	3.43	6.71	11.24	3.39	7.08	11.41
	Newspaper	3.49	7.15	13.11	3.61	7.09	12.22
	GT_Fly	3.75	7.87	12.60	3.78	8.07	13.26
	PHall2	3.62	7.42	11.58	3.61	7.58	11.89
	PStreet	3.38	7.44	11.29	3.82	8.10	14.13
	Dancer	3.52	7.24	10.71	3.78	7.78	13.14
	Shark	3.15	6.56	9.80	3.71	7.73	12.49
	Average	**3.47**	**7.12**	**11.39**	**3.68**	**7.63**	**12.46**

For showing higher details two evaluations were selected, Fig. 5(a) and (b) show the speedup of TITAN and uniform-sized tiles per frame of Undo_Dancer video sequence using Quantization Parameter QP=(30, 39) (texture, depth), where texture is presented in Fig. 5(a) and depth is presented in Fig. 5(b). The secondly selected evaluation is presented in Fig. 5(c) and (d), where Shark video sequence was evaluated with QP=(25, 34). In both evaluations, the speedup of TITAN and uniform-sized tiles are similar for the first frame since TITAN requires the encoding time spent in the previous frame for adapting its balancing. However, TITAN immediately acted to enhance the workload balancing, allowing a better workload distribution into different tiles and consequently obtaining a higher speedup than the uniform-sized tiles, as Fig. 5 shows.

Since this work is the pioneer considering speedup tiles balancing for 3D-HEVC encoding, Table III presents comparisons with the related work [9], which was proposed for 2D HEVC encoding context, considering the average BD-rate and speedup results for 2×2, 3×3 and 4×4 tile distribution. For a fair comparison, our results consider only texture results.

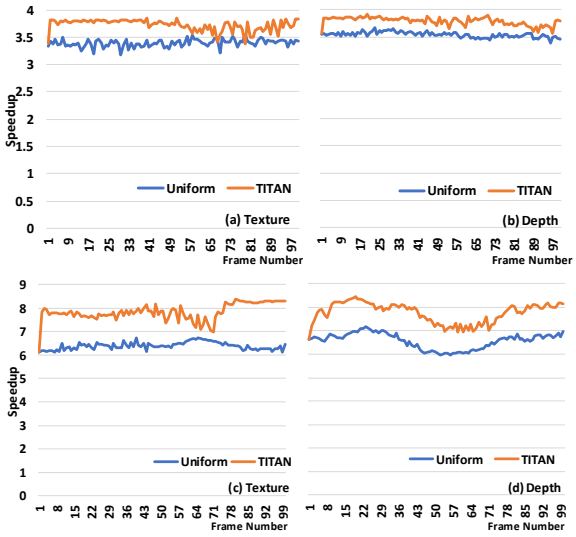

Fig. 5 – Speedup results for Undo_Dancer video sequence with QP=(30, 39) for 2×2 tile distribution with (a) texture and (b) depth maps, and for Shark video with QP=(25, 34) for 3×3 tile distribution with (c) texture and (d) depth maps.

Table III – Comparison of texture BD-rate and speedup with related work.

Work	2×2		3×3		4×4	
	BD-rate	Speedup	BD-rate	Speedup	BD-rate	Speedup
Our	1.4%	3.75	2.6%	7.71	4.9%	13.62
[9]	1.4%	3.62	3.0%	7.32	5.5%	12.57

Table III shows that for all distribution of tiles evaluated our work surpass the speedup when compared to the [9]. When considering the BD-rate, this work achieves better results for 3×3 and 4×4 and the same impact for 2×2 tile distribution. Thus, this work can provide a higher tradeoff for parallelization encoding with lower impact on encoding efficiency.

V. CONCLUSIONS

This paper presented the Tile Timing-Aware balancing (TITAN) algorithm aiming to speed up the 3D-HEVC intra encoding. Motivated by the similar encoding effort of neighbor frames correlated tile balancing, TITAN controls the tiles boundaries with the focus of increasing the workload balancing among tiles. Therefore, higher speedup can be achieved when the execution is distributed into multiple homogeneous encoding cores. Software evaluations demonstrated that the TITAN algorithm reaches a higher speedup than uniform-sized tiles without affecting the encoding efficiency. Besides, we demonstrated that its results could be applied also in HEVC since it achieved better results than related works regarding both speedup and BD-rate in texture coding. We also plan to expand the TITAN algorithm for inter-frames as future work.

ACKNOWLEDGMENT

This paper was performed in cooperation with Hewlett-Packard Brazil Ltda. using incentives of Brazilian Informatics Law (Law no 8.248 of 1991). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, and also by the Brazilian research support agencies CNPq and FAPERGS.

REFERENCES

[1] G. Sullivan, J. Ohm, W. Han, T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Transactions on circuits and systems for video technology (TCSVT)*, v. 22, n. 12, pp. 1649-1668, Dec. 2012.

[2] G. Tech, Y. Chen, K. Muller, J. Ohm, A. Vetro, Y. Wang, "Overview of the Multiview and 3D extensions of High Efficiency Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, v. 26, n. 1, pp. 35-49, Jan. 2016.

[3] P. Kauff, N. Atzpadin, C. Fehn, M. Muller, O. Schreer, A. Smolic, R. Tanger, "Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability," *Signal Processing: Image Communication*, v. 22, n. 2, pp. 217-234, Feb. 2007.

[4] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," *Stereoscopic Displays and Virtual Reality Systems (SPIE)*, v. 5291, pp. 93-104, May 2004.

[5] C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, T. Schierl, "Parallel scalability and efficiency of HEVC parallelization approaches", *Transactions on circuits and systems for video technology (TCSVT)*, v. 22, n. 12, pp. 1827-1838, Dec. 2012.

[6] M. Tang; X. Chen; J. Gu; Y. Han; J. Wen; S. Yang, "Accelerating HEVC Encoding Using Early-Split", *IEEE Signal Processing Letters*, v. 25, n. 2, pp. 209-2013, Feb. 2018.

[7] C. Yan; Y. Zhang; J. Xu; F. Dai; L. Li; Q. Dai; F. Wu, "A Highly Parallel Framework for HEVC Coding Unit Partitioning Tree Decision on Many-core Processors", *IEEE Signal Processing Letters*, v. 21, n. 5, pp. 573-576, Feb. 2014.

[8] C. Blumenberg; D. Palomino; S. Bampi; B. Zatt, "Adaptive content-based Tile partitioning algorithm for the HEVC standard", *Picture Coding Symposium (PCS)*, 2013.

[9] I. Storch; D. Palomino; B. Zatt; L. Agostini, "Speedup-aware history-based tiling algorithm for the HEVC standard", *IEEE International Conference on Image Processing (ICIP)*, 2016.

[10] M. Koziri, P. K. Papadopoulos, N. Tziritas, A. N. Dadaliaris, T. Loukopoulos, S. U. Khan, C.-Z. Xu, "Adaptive Tile Parallelization for Fast Video Encoding in HEVC*", IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016.

[11] G. Sanchez, J. Silveira, L. Agostini, C. Marcon, "Performance Analysis of Depth Intra Coding in 3D-HEVC", *Transactions on circuits and systems for video technology (TCSVT)*, 2018.

[12] 3D-HEVC Test Model. Available at: https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-16.0/, access in Sep. 2017.

[13] K. Muller, A. Vetro, "Common Test Conditions of 3DV Core Experiments", ISO/IEC JTC1/SC29/WG11 MPEG2011/ N12745, Jan. 2014.

[14] G. Bjontegaard, "Calculation of Average PSNR Differences between RD-Curves," VCEG Meeting, pp. 1-5, 2001.