3D-HEVC DMM-1 Parallelism Exploration Targeting Multicore Systems

[1,2,3]Gustavo Sanchez, [2,4]Luciano Agostini, [2]Leonel Sousa, and [1]César Marcon
[1]*Pontifical Catholic University of Rio Grande do Sul – Porto Alegre, Brazil*
[2]*INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Lisbon, Portugal*
[3]*IF Farroupilha – Alegrete, Brazil*
[4]*Video Technology Research Group (ViTech) – Federal University of Pelotas (UFPel) – Pelotas, Brazil*
gfsanchez@acad.pucrs.br, agostini@inf.ufpel.edu.br, las@inesc-id.pt, cesar.marcon@pucrs.br

***Abstract*— This paper proposes the 3D-HEVC Depth Modeling Mode One (DMM-1) parallelization in a multicore environment. Mainly data parallelism is explored. Experimental results with a four-core processor extended to eight logic cores through hyper-threading show that the proposed parallel version of DMM-1 achieves a speedup of 5.8. Besides, it executes 212 times faster when combined with the Simplified Edge Detector (SED) and the Gradient-Based Mode One Filter (GMOF), which are timesaving heuristics designed by related works. This is a pioneer work exploring the DMM-1 parallelism in a multicore scenario, which can be extended to massive parallel environments with even more impressive results.**

Keywords—Depth Modeling Modes, 3D-HEVC, Multicore, Parallelism Exploration.

I. Introduction

Significant advances in video coding were obtained with the High Efficiency Video Coding (HEVC) standardization, making possible to reach a much higher compression rate when compared with the previous standards, while maintaining a similar video quality [1]. These advances were also extended to different digital video sources such as: (i) 3D video coding, through the 3D-HEVC extension [2]; (ii) Multiview Coding, using the MV-HEVC extension; (iii) Screen Content, through the HEVC-SCC, and (iv) Format Range Extension, employing the RExt-HEVC extension [3]. The encoding efficiency gain provided by these new extensions has as a counterpart, the increasing of the encoder computational effort, thus demanding new techniques to speedup the encoding process.

In the 3D-HEVC, one of the main reasons for raising the computational effort is the use of the Multiview Video plus Depth (MVD) data format [4], because each texture frame is associated with a depth map and both must be encoded and packed into the bitstream. Besides, MVD allows synthesizing a dense set of intermediary high-quality virtual views at the decoder, using Depth Image Based-Rendering (DIBR) [5].

The 3D-HEVC texture frames represent conventional 2-D scenes, and the depth maps provide the distance between the camera and the objects in the scene. An example of a texture frame and its associated depth map is presented in Fig. 1 [6]. While texture frames have smooth transitions between pixels, depth maps are composed of homogeneous regions and sharp edges. The homogeneous regions correspond to the background of the scene and the body of the objects, while the sharp edges are located on the border of the objects. The algorithms designed for texture coding in HEVC were inherited for depth maps coding. However, these algorithms are not specialized to explore the depth maps characteristics and more efficient solutions can be obtained if specialized algorithms are designed. Therefore, new encoding tools were inserted in the 3D-HEVC to improve the depth maps encoding efficiency. These new tools can be used as an alternative to the HEVC texture tools, which are also available. Most of these advanced tools are related to intra-frame prediction, such as Depth Modeling Modes 1 and 4 (DMM-1 and DMM-4) [7], Segment-wise Direct Component Coding (SDC) [8], and Depth Intra Skip (DIS) [9].

(a) (b)

Fig. 1. Kendo video sequence (a) texture view and (b) depth map [6].

In the work [10], the intra-frame prediction encoding steps effort was analyzed. Among the new tools, it is notable the impact of SDC and DMM-1 on the encoder computational effort. Several works such as [11]-[14] already proposed techniques to accelerate the DMM-1 execution. However, all of them impose significant encoding efficiency losses. Therefore, it is important to investigate innovative solutions to accelerate the 3D-HEVC depth maps coding without impact in the encoding efficiency. In this scenario, the parallel execution in multicore systems could be an effective alternative. The DMM-1 algorithm can be parallelized using different granularities on a multicore CPU. However, the SDC execution cannot be easily parallelized since it contains data dependencies with the entropy encoder, which has a sequential and irregular nature [15]. Thus, SDC is not a good target for parallelism exploration.

This paper presents two approaches to explore the DMM-1 parallelism over multicore systems using OpenMP [16]: block granularity and pattern granularity. Both approaches for 3D-HEVC encoding are neutral in what concerns encoding efficiency. Eight levels of parallelism were evaluated, using one

978-1-5386-7431-4/18/$31.00 ©2018 IEEE

to eight threads to process the DMM-1. The original DMM-1 algorithm and two heuristics to accelerate the DMM-1 calculations available at the literature were also used in this investigation aiming to demonstrate that simplified DMM-1 versions can obtain benefits if the encoding is distributed among the available cores. The two heuristics are the Simplified Edge Detector (SED) [11] and the Gradient-Based Mode One Filter (GMOF) [12]. The SED and GMOF heuristics are fully compliant with the 3D-HEVC standard.

The evaluation environment used in this paper is a CPU containing four physical cores. However, the obtained results could be extended to other multicore systems, such as Graphics Processor Units (GPU) systems, Multiprocessor System-on-Chip (MPSoC) or dedicated hardware designs. To the best of our knowledge, this is a pioneer work on parallelism exploration to develop efficient DMM-1 encoding.

The remaining of this paper is organized as follows. The basics of the 3D-HEVC depth maps intra-frame prediction is presented in Section II. Section III describes the DMMs, mainly focusing on the DMM-1 algorithm. Section IV proposes the parallel algorithms to speed up the computation of DMM-1. Our experimental setup is presented in Section V and the results and its respective discussion in Section VI. Section VII concludes this paper.

II. 3D-HEVC Depth Maps Intra-Frame Prediction

Fig. 2 shows the 3D-HEVC depth maps intra-frame prediction dataflow. All depth blocks should be evaluated using the conventional HEVC intra-frame prediction, DIS, DMM-1, and DMM-4. The first three tools generate a subset of modes that are inserted into a list, called RD-list. These modes use Transform-Quantization (TQ) and SDC tools, while the DIS does not use TQ and SDC steps. Finally, the produced results are entropy encoded, and then the RD-cost is calculated to define which mode is inserted in the final encoded stream.

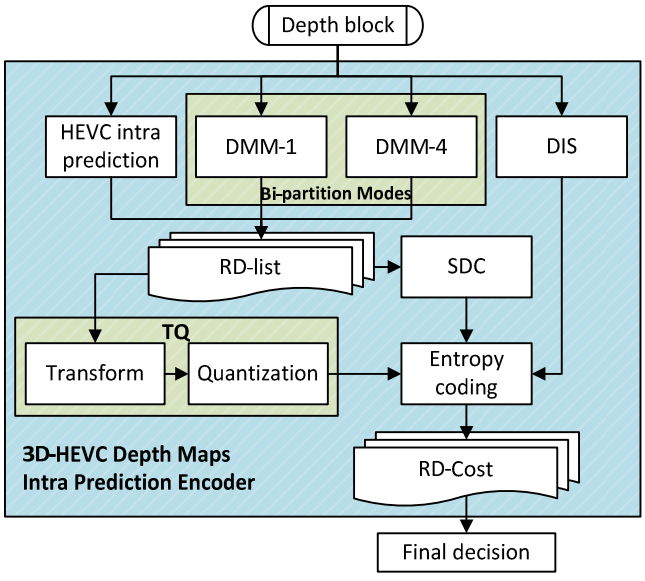


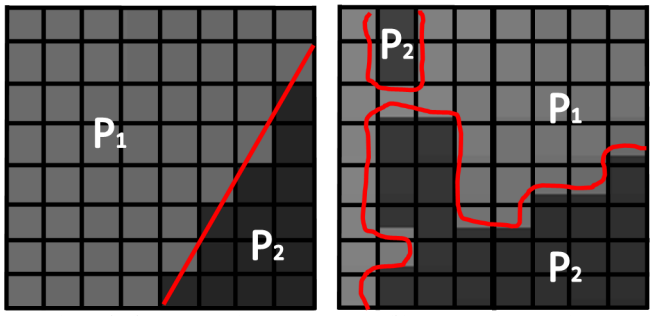
Fig. 2. 3D-HEVC depth maps intra-frame prediction dataflow model [10].

The HEVC intra-frame prediction was inherited from the texture coding, without any modification. It has 35 modes that can be evaluated: planar mode, Direct Component (DC) mode and, 33 directional modes [13]. DIS is a new tool defined by the 3D-HEVC. Since most of the depth map information corresponds to smooth areas, DIS encoding mode was mainly focused on reducing the bitrate of these areas by not packing the residues in the bitstream. The SDC was proposed by the 3D-HEVC to obtain higher efficiency encoding homogeneous blocks or edges regions divided into two homogeneous regions. DMMs are focused on obtaining a better prediction in edges regions, as detailed in Section III.

III. The Depth Modeling Modes

The Depth Modeling Modes (DMMs) are composed of the DMM-1 and DMM-4 encoding tools. DMM-1 divides the encoding block using wedgelet patterns, as presented in Fig. 3(a), where a wedgelet is a straight line that divides the encoding block into two regions. While DMM-1 assumes only the predefined wedgelets patterns defined by the 3D-HEVC standard, DMM-4 divides the block into two regions using contours that can assume arbitrary patterns and can consist of several parts, as illustrated in Fig. 3(b). Both DMMs are applied at different block sizes, ranging from 4×4 to 32×32.

DMM-4 produces the segmentation pattern using the texture data. Since the texture data is also available at the decoder, the DMM-4 decoder can use the same algorithm than that used by the encoder to generate the pattern. Then, the pattern is not transmitted, reducing the amount of data to encode.



(a) Wedgelet (DMM-1) (b) Contour (DMM-4)

Fig. 3. Example of a wedgelet and a contour segmentation.

Fig. 4 describes the DMM-1 algorithm-encoding flowchart, which is composed of the following stages: (i) Main, (ii) Refinement, and (iii) Residue.

The Main Stage evaluates an initial predefined wedgelet set by searching for the wedgelet that produces the lowest distortion at the synthesized views, in comparison to the original encoding samples. To find the best distortion, the encoding block is mapped into the binary pattern defined by each wedgelet, and the average value of each region is computed according to this mapping.

Subsequently, the DMM-1 algorithm generates the predicted depth block using the average value of each region in the Prediction Step. The Synthesized View Distortion Change (SVDC) [17] computes the distortion of a synthesized block in comparison to the synthesizing texture views using the original depth block values in the Distortion step. The wedgelet that leads to the lowest distortion is selected.

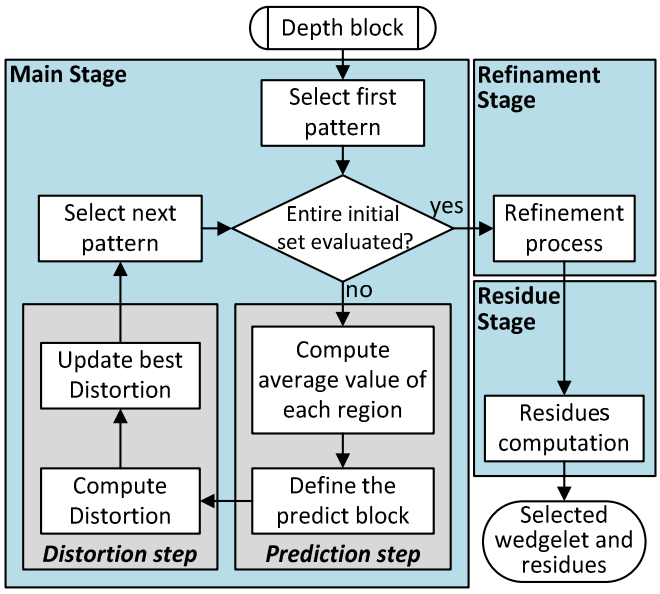


Fig. 4. DMM-1 encoding algorithm.

The Refinement Stage evaluates the distortion of up to eight wedgelets (with slight differences from the best wedgelet that was selected in the Main Stage) using SVDC. The wedgelet with the lowest distortion is chosen as the best one to encode the current depth map block, and the residues of this block are calculated in the Residue Stage. Therefore, this wedgelet is inserted in the RD-list with the residues of this block to be used in the TQ and SDC evaluations.

IV. Proposed Parallel Approaches

Two parallel approaches at a different level of granularity were investigated in this work to handle with the DMM-1 calculations: (i) block granularity and (ii) pattern granularity.

Fig. 5(a) illustrates the main idea of the block granularity approach. Since DMM-1 evaluation does not include any data dependencies among neighbor blocks, the computational effort to encode more than one block at the same time is distributed into several cores, one block size per time. Each thread is responsible for the whole DMM-1 encoding of a depth block, requiring to apply the main, refinement and residue stages. Moreover, since DMM-1 requires the evaluation of several wedgelets patterns, then the other pattern granularity approach herein considered distributes the evaluation of different DMM-1 patterns along different threads, as illustrated in Fig. 5(b). In this granularity, each wedgelets in the main stage is assigned to a different thread for distributing the computational effort among different CPU cores. After computing the distortion of all wedgelets, then the threads are synchronized, and the best distortion is computed. In the refinement stage, the wedgelets patterns are also distributed among different threads.

Considering that exist several simplifications of the DMM-1 algorithm, the SED [11] and GMOF [12] heuristics were applied over the block granularity approach to evaluating the gains that DMM-1 simplifications could bring in this multicore scenario. The SED algorithm skips the DMMs evaluation for homogeneous blocks since the DMMs tend not to be selected in this scenario. This pre-processing can reduce the DMM-1 computational effort about 93.4% (speedup of 15.1) with a degradation of 0.94% in BD-rate [11]. The GMOF directly simplifies the DMM-1 evaluation by applying a gradient filter in the border of the encoding block to select the most promising wedgelets to segment the block. Therefore, several wedgelets evaluations are skipped with this algorithm (reducing the DMM-1 computational effort in 66.9%, achieving a speedup of 3.0) with minor degradation in the encoding efficiency (0.33% in BD-rate, on average) [12]. The evaluations of these two heuristics were initially done through a sequential processing using the 3D-HEVC Test Model (3D-HTM) [18], producing the results referred previously. This work presents the performance achieved with these heuristics when parallel algorithms are executed on the multicore system.

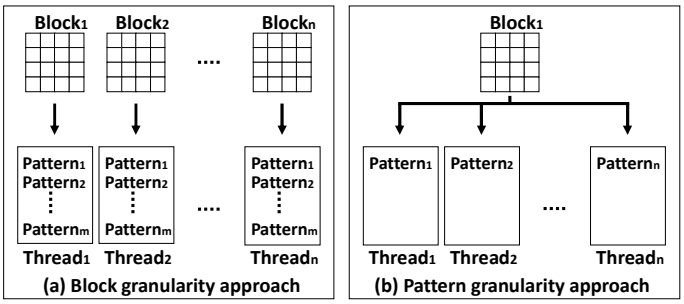


Fig. 5. Parallelism exploration: (a) block granularity and (b) pattern granularity.

V. Experimental Setup

The 3D-HTM [18] was developed mainly to evaluate the efficiency of the encoder tools; it is not suitable for real-time applications since the associated execution time was far from being optimal. Therefore, to assess the multicore parallel execution of DMM-1, we designed in C++ a software that implements the DMM-1 and uses OpenMP to explore the parallelism. Besides the original algorithm, two other versions were implemented, one using the SED heuristic and other using the GMOF heuristic.

The designed software was developed to encode the central view of the eight videos available at Common Test Conditions (CTC) for 3D experiments [19]. The software inputs are the raw depth maps data and the reconstructed texture video obtained from 3D-HTM 16.2. The developed software provides the residual data and identification of the selected DMM-1 pattern.

The encoding time spent during each evaluation was registered considering the execution time and the overhead produced by reading the frames from memory. The encoding time of all experiments presented in this paper considered only the DMM-1 execution. All evaluations were done using an Intel Core i7 4770K with 4 cores (8 threads) running at 3.5 GHz with a 32 GB DDR3 memory. Although we performed the evaluations using a four-core CPU, the presented experiments aim to demonstrate that several encoding blocks can be distributed among a diverse number of cores to accelerate the encoding process. Besides, the proposed methodology could be used in systems with a higher number of cores, increasing the speedup achieved.

VI. Results and Discussion

This section presents the experimental results obtained when the proposed parallel algorithms are applied. Fig. 6 displays a comparison between the two previously presented parallelism exploration approaches. This comparison considered the DMM-1 execution time in a single thread and parallelized in eight threads with two granularity levels: block and pattern. Fig. 6 shows the average times per frame for the eight CTC sequences.

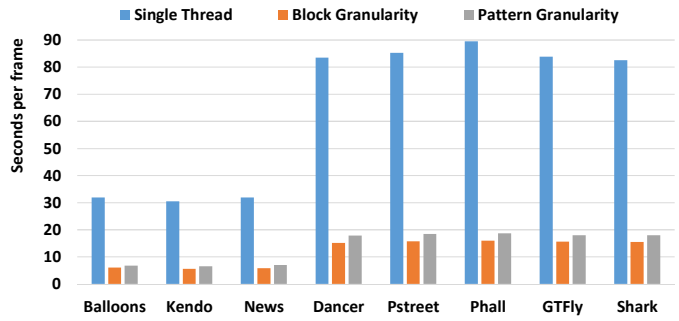


Fig. 6. Time per encoded frame considering the evaluated approaches.

The average times required to process a frame with 1024×768 pixels (Balloons, Kendo and News sequences) using the block granularity, pattern granularity, and only one thread were 5.5, 6.8 and 31.4 seconds, respectively. Considering frames with 1920×1088 pixels (Dancer, Pstreet, Phall, GTFly and Shark sequences), the average processing time per frame was reduced from 85.0 to 14.7 and 18.3 seconds, when using block and pattern granularities over four cores, respectively.

The average percentage gains over the single thread version are similar for the two evaluated resolutions, where the block granularity reached a global average speedup of about 6 and the pattern granularity reached an average speedup about 5. Both approaches greatly reduce the required time per frame. Since the best results were always achieved with the block granularity approach, then we adopt this approach for the next experiments.

The second evaluation presented in this paper was done varying from one to eight the number of executing threads using the block granularity approach. We measured the execution time for each of these cases and compared it with the time required by a single thread. Results of these evaluations are presented in Fig. 7(a), namely the achieved speedup taking as the reference the single-thread execution. Notice that when using eight threads, our solution was able to reach an average speedup of 5.7 times. Fig. 7(b) shows that the gains grow slower using five or more threads (e.g., increasing from 4 to 5 threads, the speedup increases only 11.4%). This behavior is explained by the fact that the system has only four physical cores and, when using more than four threads, the hyper-threading allows the usage of up to 8 logic cores, but the speedup is reduced. However, one can notice that the gain does not saturate, demonstrating that higher speedups could be attained when using a higher number of cores.

The next experiment was done integrating the SED [11] and GMOF [12] into the implemented software, then, three new versions were generated: (i) using SED; (ii) using GMOF; and (iii) using SED and GMOF together. The average speedups are presented in Fig. 8, where a logarithmic scale was used to display better the reached results. The three new versions presented impressive speedup gains when compared with the multithread version using block granularity, independently of the number of used threads. The average speedup is 42 times higher than the reached with the block granularity version when SED and GMOF are used together. It is important to highlight that our parallel implementation introduces the same encoding efficiency loss as using those timesaving algorithms in a single thread execution, therefore the evaluated solutions are capable of accelerating the DMM-1 execution significantly without inserting additional encoding loss than the single thread execution.

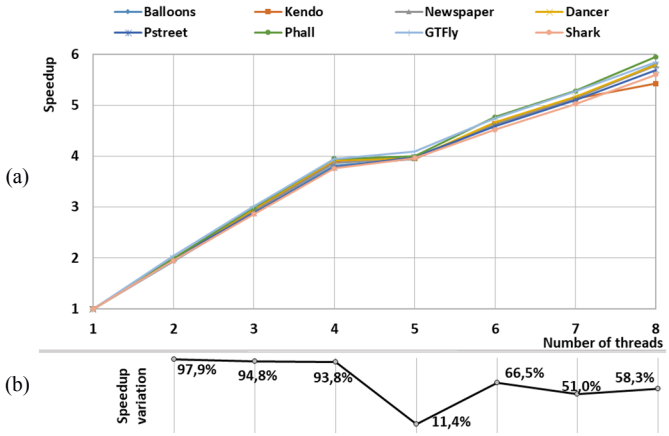


Fig. 7. (a) Speedups according to the number of threads for the block granularity and (b) speedup variation according to the increase of the threads.

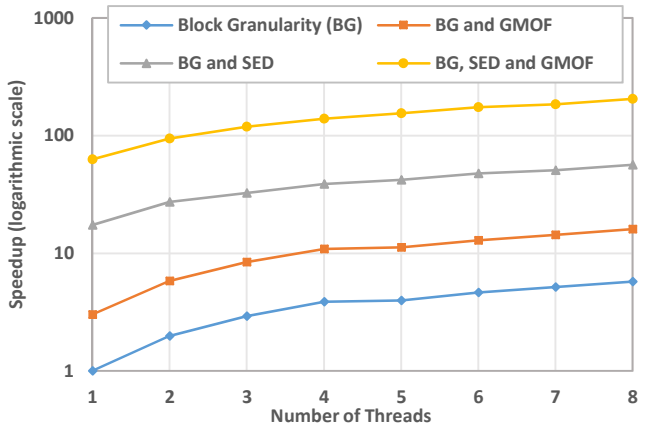


Fig. 8. Speedups according to the number of threads for the block granularity with SED and GMOF algorithm.

Table I summarizes the average time per encoded frame reached when 1920×1088 videos are processed using eight threads, by taking the results of all previously discussed evaluations. The reached speedup over the single thread version are also presented. The presented results firstly show that the multithread exploration brings expressive reductions in processing time to process DMM-1. The speedup without the use of any additional heuristic are of 5.8 times when compared with the single thread version. When the SED and GMOF heuristics are used together with the multithread implementation, the speedup are even more expressive, varying

from 16 times to 212.3 times, when compared with the single thread implementation. In these cases, a little encoding efficiency drop is also observed, with a degradation of less than 2% in BD-rate caused by the SED and GMOF heuristics [20].

TABLE I. SUMMARY OF REACHED RESULTS FOR 1920×1088 VIDEOS USING EIGHT THREADS.

Implementation		Time per frame	
		Average (seconds)	Speedup (times)
Single Thread		85.0	-
Eight Threads	Pattern Granularity	18.3	4.6
	Block Granularity	14.7	5.8
	Block Granularity with GMOF	5.3	16.0
	Block Granularity with SED	1.3	65.4
	Block Granularity with GMOF and SED	0.4	212.5

Finally, it is important to emphasize that the method proposed in this work was implemented in a four-core system, but the parallelism exploration could be extended to massively parallel systems, like GPUs, MPSoCs or dedicated hardware, where the proposed approach could achieve much higher speedups than those obtained herein.

VII. CONCLUSIONS AND FUTURE WORK

This paper proposes to exploit parallelism for computing the Depth Modeling Modes One (DMM-1) in a multicore scenario. Among the most time-consuming steps in 3D-HEVC depth maps intra-frame prediction, DMM-1 presents the best opportunities for depth parallelism exploration, since it does not present data dependencies among neighbor blocks and encoding tools. However, DMM-1 requires one of the highest computational efforts among all 3D-HEVC encoding tools. Therefore, this paper presents a DMM-1 multithread implementation employing OpenMP to investigate the DMM-1 behavior when running on parallel platforms. Two heuristics (SED and GMOF) were also included in the presented experiments to observe their behavior in this parallel environment. Evaluation results show that using a hyper-threading processor with four cores the parallel version can accelerate the DMM-1 encoding execution up to 5.8 times. The gains increase significantly when the SED and GMOF heuristics are used, reaching, in the best case, a speedup of 212 times when compared with the single thread version. Since the presented speedup curves did not present a saturation behavior, it is possible to conclude that the use of the proposed approach in a massively parallel scenario (using GPU, for example), could reach more significant speedups.

ACKNOWLEDGMENT

This paper was achieved in cooperation with Hewlett-Packard Brazil Ltda. using incentives of Brazilian Informatics Law (Law nº 8.248 of 1991). Authors also would like to thanks CAPES (processes 88881135737/2016-01 and 88881119481/2016-01), CNPq and FAPERGS Brazilian research agencies to support the development of this work. This work was also supported by the National Funds through Fundação para a Ciência e a Tecnologia (FCT) under Project UID/CEC/50021/2013.

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] K. Muller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F. Rhee, G. Tech, M. Winken, T. Wiegand. "3D High-Efficiency Video Coding for Multi-View Video and Depth Data," *IEEE Transactions on Image Processing* (TIP), v. 22, n. 9, pp. 3366-3378, Sep. 2013.

[3] G. Sullivan, J. Boyce, Y. Chen, J. Ohm, C. Segall, A. Vetro, "Standardized Extensions of High Efficiency Video Coding (HEVC)," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1001-1016, Dec. 2013.

[4] Y. Zhao, C. Zhu, Z. Chen, D. Tian, L. Yu, "Boundary Artifact Reduction in View Synthesis of 3D Video: From Perspective of Texture-Depth Alignment," *IEEE Transactions on Broadcasting*, v. 57, n. 2, pp. 510-522, Jun. 2011.

[5] 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.

[6] Tanimoto Lab. Available at: http://www.tanimoto.nuee.nagoya-u.ac.jp/, access in Sep. 2017.

[7] P. Merkle, K. Muller, D. Marpe, T. Wiegand, "Depth Intra Coding for 3D Video based on Geometric Primitives," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, v. 26, n. 3, pp. 570-582, Feb. 2015.

[8] H. Liu, Y. Chen, "Generic segment-wise DC for 3D-HEVC depth intra coding," in *Proc. IEEE International Conference on Image Processing (ICIP)*, pp. 3219-3222, 2014.

[9] J. Lee, M. Park, C. Kim, "*3D-CE1: Depth Intra Skip (DIS) Mode*," document JCT3V-K0033, Geneva, Switzerland, Feb. 2015.

[10] G. Sanchez, L. Agostini, C. Marcon, "3D-HEVC depth maps intra prediction complexity analysis", in Proc. *IEEE International Conference on Electronics, Circuits & Systems* (ICECS), pp. 1-4, 2016.

[11] G. Sanchez, M. Saldanha, G. Balota, B. Zatt, M. Porto, L. Agostini, "Complexity reduction for 3D-HEVC depth maps intra-frame prediction using simplified edge detector algorithm," *International Conference on Image Processing (ICIP)*, pp. 3209-3213, 2014.

[12] G. Sanchez, M. Saldanha, B. Zatt, M. Porto, L. Agostini, "A complexity reduction algorithm for depth maps intra prediction on the 3D-HEVC," *IEEE Visual Communication and Image Processing Conference (VCIP)*, 2014.

[13] J. Lainema, F. Bossen, W. Han, J. Min, K. Ugur, "Intra Coding of the HEVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, v. 22, n. 12, pp. 1792-1801, Dec. 2012.

[14] Q. Zhang, Q. Yang, Y. Chang, W. Zhang, Y. Gan. "Fast intra mode decision for depth coding in 3D-HEVC," *Multidimensional Systems and Signal Processing*, pp. 1-24, 2016.

[15] D. Souza, A. Ilic, N. Roma, L. Sousa, "GHEVC: An efficient HEVC decoder for graphics processing units," *IEEE Transactions on Multimedia*, v. 19, n. 3, Mar. 2017.

[16] L. Dagum, R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Computational Science and Engineering*, v. 5, n. 1, pp. 46-55, Jan.-Mar. 1998.

[17] G. Tech, H. Schwarz, K. Müller, T. Wiegand, "3D video coding using the synthesized view distortion change," in *Proc. Picture Coding Symposium (PCS)*, pp. 25-28, May 2012.

[18] Y. Chen, G. Tech, K. Wegner, S. Yea. "Test Model 10 of 3D-HEVC and MV-HEVC," ISO/IEC *JTC1/SC29/WG11*, Strasbourg, Oct. 2014.

[19] D. Rusanovskyy, K. Muller, A. Vetro. "Common Test Conditions of 3DV Core Experiments," ISO/IEC JTC1/SC29/WG11 MPEG2011/N12745, Geneva, Jan. 2013.

[20] G. Sanchez et. al "DMMFast: a complexity reduction scheme for three-dimensional high-efficiency video coding intraframe depth map coding," Journal of Electronic Imaging, v. 24, n. 2, pp. 023011-023011, Mar. 2015.