J Real-Time Image Proc (2017) 13:71–83
DOI 10.1007/s11554-016-0609-8

SPECIAL ISSUE PAPER

Real-time scalable hardware architecture for 3D-HEVC bipartition modes

Gustavo Sanchez[1] · **César Marcon**[1] · **Luciano Agostini**[2]

Received: 16 December 2015 / Accepted: 2 June 2016 / Published online: 16 June 2016

Abstract This article presents a real-time scalable hardware architecture for the bipartition modes of 3D high-efficiency video coding (3D-HEVC) standard, which includes the depth modeling modes 1 (DMM-1) and 4 (DMM-4). A simplification of the DMM-1 algorithm was done, removing the refinement step. This simplification causes a small BD-rate increase (0.09 %) with the advantage of better using our hardware resources, reducing the necessary memory required for storing all DMM-1 wedgelet patterns by 30 %. The scalable architecture can be configured to support all the different block sizes supported by the 3D-HEVC and also to reach different throughputs, according to the application requirements. Then, the proposed solution can be efficiently used for several encoding scenarios and many different applications. Synthesis results considering a test case show that the designed architecture is capable of processing HD 1080p videos in real time, but with other configurations, higher resolutions are also possible to be processed.

Keywords 3D-HEVC · Bipartition modes · Scalable architecture · Hardware design

✉ Gustavo Sanchez
gustavo.sanchez@acad.pucrs.br

César Marcon
cesar.marcon@pucrs.br

Luciano Agostini
agostini@inf.ufpel.edu.br

[1] PPGCC – PUCRS, Porto Alegre, Brazil

[2] GACI - PPGC – UFPEL, Pelotas, Brazil

1 Introduction

The Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) is a group of experts from ITU-T and ISO/IEC, which was established to work on multiview and 3D video coding extensions of high-efficiency video coding (HEVC). Due to the increase in 3D video coding usage, the JCT-3V spends significant effort in research and development to extend the HEVC standard to 3D video applications [1], which reduces the bandwidth for a 3D video transmission or storage with similar quality [2]. The 3D-HEVC standard was finalized in February 2015 by JCT-3V. It uses the most advanced features provided by HEVC and proposes many new features to explore 3D video characteristics.

A multiview video sequence consists of many views captured from multiple cameras, displaced very close to each other, to cover a different portion of a scene to support 3D applications [3]. Many coding features have already been included in the previous 3D standard [4] to encode videos from multiple cameras efficiently. However, 3D-HEVC uses even more advanced texture coding tools such as neighboring block-based disparity vector derivation [5] [6], inter-view motion prediction, inter-view residual prediction [7] and illumination compensation [8].

Another key factor for 3D-HEVC efficiency (regarding the quality of compression) is the adoption of the multiview plus depth (MVD) [9] representation to encode and transmit an enormous amount of data required by 3D video applications. In MVD, each texture view is associated with a depth map. The same camera captures the texture images and depth maps, which provide geometrical information according to the objects distance from the camera [10]. Eight-bit samples in gray shades compose these maps, where the closer the object is from the camera, the lighter is

the shade of gray will be used to represent the object. Figure 1 presents a (a) texture view and its associated (b) depth map extracted from *Newspaper_CC* video sequence.

The motivation for MVD usage is to reduce the bandwidth for a 3D video transmission. Because in the decoder, it is possible to synthesize virtual views [11] interpolating texture and depth data with the use of techniques such as depth-image-based rendering (DIBR) [9], these techniques allow synthesizing a dense set of texture views of the scene [12] and, consequently, to reduce the number of transmitted texture views. Figure 2 illustrates a nine-view application, which represents the advantage of using MVD format.

Traditionally, 3D standards would encode and transmit all nine texture views, while when using MVD format only a subset of those texture views (e.g., three texture views) together with their depth maps is encoded and transmitted. The 3D video decoder is capable of generating virtual views located in any position between any two encoded views (e.g., views 2 to 4 and 6 to 8 in Fig. 2), which allows considerable bandwidth reduction. This gain is important since each texture frame must be represented using three matrixes, one for each color component, and the depth map is represented using only one matrix, as previously discussed.

The quality of the depth maps is crucial to allow the generation of virtual views with high quality, and this is a significant challenge in this scenario. Figure 2 shows that depth maps contain characteristics that contrast with texture frames; i.e., they contain large areas of constant values (background or body of objects) and sharp edges (border of objects) [13], while texture frames have smooth transitions between its pixels. Therefore, using only traditional 2D video coding techniques in depth maps compression introduces blocking artifacts, mosquito noise and edge blurring [14] if a high compression is desired.

It is important to emphasize that distortions in the depth map indirectly impact on the video quality since they are used to synthesize new texture views of the same scene [15, 16]. Then, it is important to encode as precisely as possible the depth maps, preserving these edges (i.e., without smoothing them) and avoiding errors in the video synthesis process.

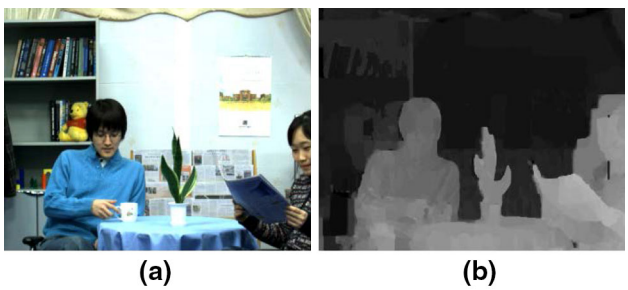

Fig. 1 **a** Texture view and its associated, **b** depth map extracted from *Newspaper_CC* video sequence

3D-HEVC considers that a depth map is encoded using all HEVC encoding tools and inserting some new tools, meaning that the depth map can be encoded using intra- or inter-prediction [17]. This paper is focused on the intra-prediction, where new tools were evaluated, and two of them were included in the standard. The original HEVC intra-prediction was not designed to explore the depth maps characteristics, such as the presence of sharp edges. Taking into account the importance of edge preservation, the JCT-3V has proposed a coding tool called bipartition modes, which should be applied in the 3D-HEVC intra-prediction to encode better depth maps. The bipartition modes were originally composed of four depth modeling modes—DMM-1 to DMM-4; and the region boundary chain—RBC [17, 18]. The final version of 3D-HEVC did not include DMM-2, DMM-3 and RBC tools [18] because they presented an adverse trade-off between complexity and coding efficiency.

There already exist many solutions focusing on simplification or hardware architectures for HEVC standard that could be extended to 3D-HEVC such as [19, 20]. Many works have already proposed simplifications specifically for 3D-HEVC such as [21–23]. However, as 3D-HEVC is still a new standard demanding much computational complexity, it still needs to obtain simplifications improvement and low-power hardware designs, mainly when requiring real-time processing 3D high definition videos running into embedded systems. The only work that we found in the literature that was focused on developing hardware for 3D-HEVC depth maps coding tools was our previous work [24], where DMM-4 architecture was designed.

Considering this scenario, this work presents a real-time scalable hardware for the bipartition modes of the 3D-HEVC. This architecture can be scaled to support all 3D-HEVC block sizes (i.e., from 4×4 to 32×32) and different throughputs.

The remainder of this article is organized as follows. Section 2 presents the 3D-HEVC depth intra-prediction algorithm and explains the traditional HEVC intra-prediction algorithm and the DMM-1 and the DMM-4 algorithms. Section 3 describes the designed bipartition modes architecture. Section 4 presents the obtained results and discussion. Finally, Sect. 5 renders the conclusions of this work.

2 3D-HEVC depth maps intra-prediction

Figure 3 illustrates the main blocks and flow of the depth map intra-prediction implemented in 3D-HEVC reference software [25]. It encloses two modules: the (1) HEVC

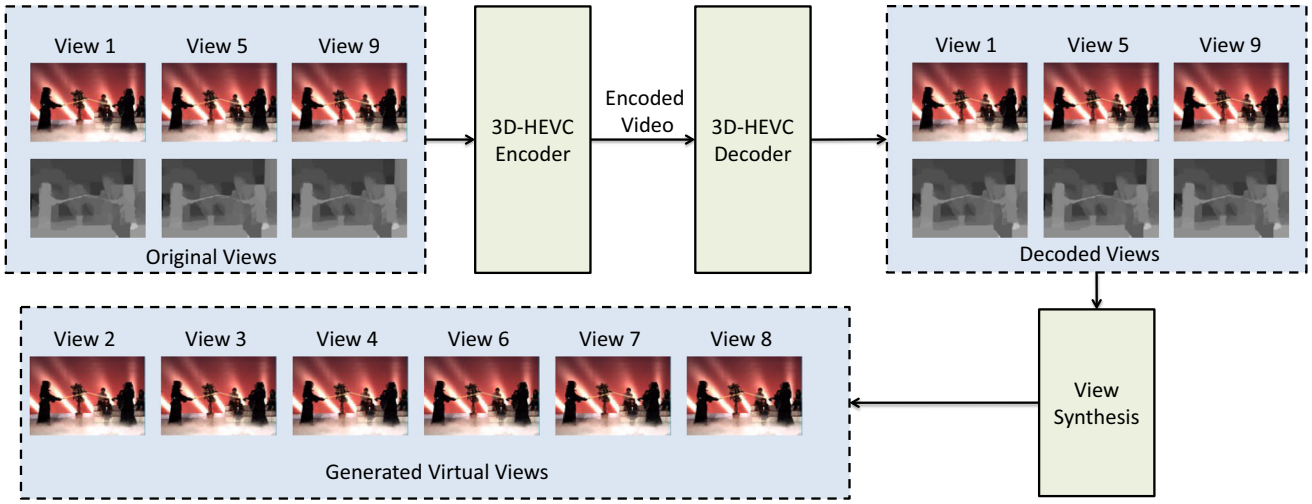

Fig. 2 Motivation of MVD format usage—tree-encoded views among with their depth maps and six virtual views generated at the decoder using view synthesis techniques

intra-prediction module that implements the same intra-algorithms used for texture videos (presented in Sect. 2.1) and the (2) bipartition modes, which focus mainly on exploring depth maps properties during the encoding process.

The bipartition modes should be evaluated in parallel with the original HEVC intra-prediction step. However, since depth coding blocks are very flat or smooth, in general, and bipartition modes obtain better results in edges of sharp transitions, 3D-HEVC depth intra-prediction algorithm employs the *fast intra-prediction mode* (proposed in [26]) to enable/disable the use of bipartition modes.

The *fast intra-prediction mode* only enables the bipartition modes encoding when the first mode in the rate–distortion list (RD-list) is not the planar mode and if the encoding block variance is higher than a predefined threshold [26], whose combination means that the encoding block does not tend to be flat or smoothing. As in flat or smoothing depth blocks, the HEVC intra-prediction obtains best results, this procedure reduces the computational complexity of intra-prediction process without reducing encoding efficiency, because it reduces significantly the percentage of bipartition modes evaluated when they are not necessary.

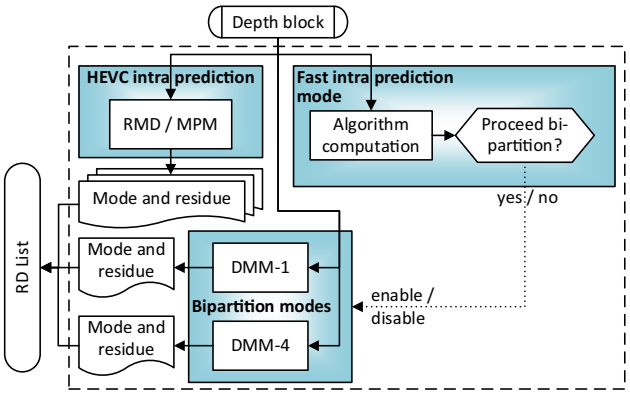

Fig. 3 Main blocks and flow of 3D-HEVC depth map intra-prediction

When the bipartition modes are enabled, the DMM-1 and DMM-4 are processed, and their results are added in the RD-list. In the following steps inside the encoder, the RD-cost is computed for all modes inside this list (HEVC intra-modes and bipartition modes).

The bipartition modes model an encoding depth block, approaching its original signal by two rectangular regions, which are represented by a constant value. Then, it is necessary a pattern containing the segmentation information for modeling a block with a bipartition mode specifying which region each sample belongs to, and the constant value that represents each region [27]. The pattern containing the segmentation information is composed of an array with $N \times N$ elements (N is the quantity of pixels that contains a side of a square image), containing 0 or 1 when the element belongs to region 0 or 1, respectively.

The 3D-HEVC bipartition modes use DMM-1 and DMM-4 algorithms, which produce wedgelet (Fig. 4) and contour (Fig. 5) [28] segmentation, respectively.

The wedgelet segmentation pattern divides the block with a straight line—Fig. 4a. As there are samples near of the straight line that are not fully inside a region, a discretization should be performed to select which region these samples belongs to, as presented in Fig. 4b.

While DMM-1 assumes only predefined patterns, the DMM-4 employs a contour segmentation that can model arbitrary patterns and even consisting of several parts (but only two regions with constant values are allowed).

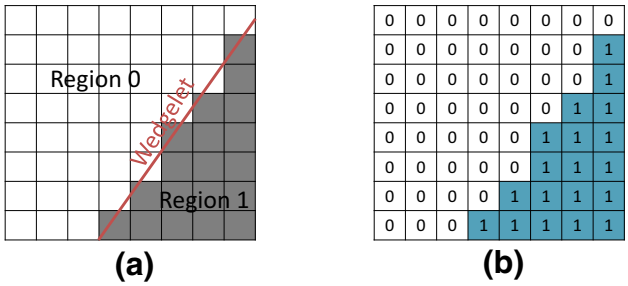

Fig. 4 Wedgelet segmentation model of a depth block: **a** pattern with region 0 and region 1 and **b** discretization with constant values

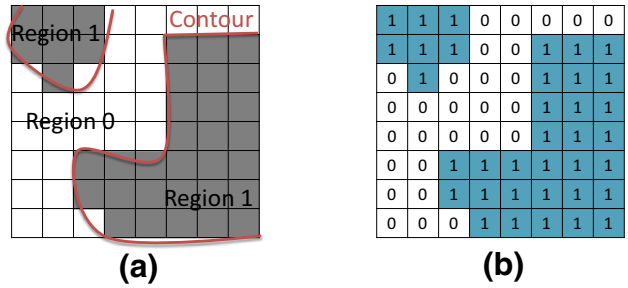

Fig. 5 Contour segmentation model of a depth block: **a** pattern with region 0 and region 1 and **b** discretization with constant values

Figure 5a exemplifies the contour segmenting a block in two regions, and Fig. 5b presents the contour signal model with the constant value of each region.

The next sections will better explain the main tools used in the depth maps intra-prediction. Section 2.1 describes the HEVC intra-prediction, and Sects. 2.2 and 2.3 detail the DMM-1 and DMM-4 segmentation algorithms, respectively.

2.1 HEVC intra-prediction

The 3D-HEVC depth map intra-prediction can use all the HEVC intra-prediction coding tools. The HEVC intra-prediction defines the planar mode, the DC mode and 33 directional modes, whose directions are presented in Fig. 6. Samples of spatially neighboring blocks are used as a reference when creating a predicted block using these modes [29].

The full RD evaluation, which performs an exhaustive calculation of all defined HEVC modes searching for the lowest RD-cost, is the most intuitive approach [30]. It is hard to deal with applications that require a real-time operation and low power dissipation when performing exhaustive approaches, needing techniques for complexity reduction and dedicated hardware design.

L. Zhao et al. [29] proposed a technique that simplifies the full RD evaluation, saves time and does not affect the encoding efficiency significantly. They proposed to create a

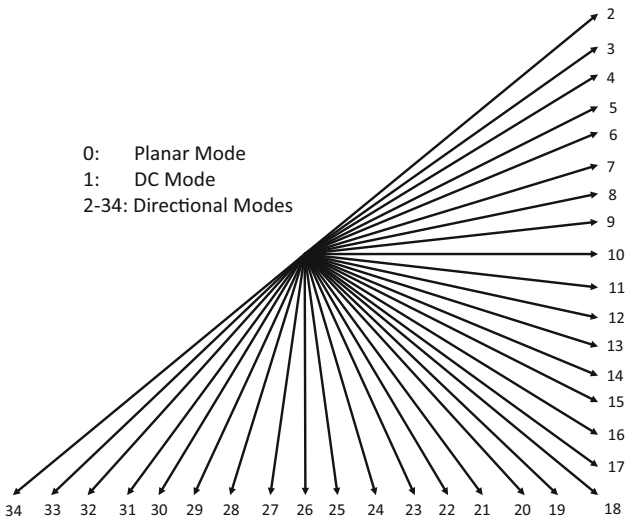

Fig. 6 HEVC intra-prediction directions

list containing few modes among all available modes and only apply RD-cost computation over them. This technique, which was adopted in HEVC reference software and consequently in 3D-HEVC intra-prediction, is described next.

This technique surrounds two algorithms: the rough mode decision (RMD) and the most probable modes (MPM). In RMD algorithm, the sum of absolute transformed differences (SATD) between the original block and the predicted one enables to evaluate all HEVC intra-modes locally (without the complete RD-cost evaluation). The algorithm orders the modes in a list by their SATDs and inserts the modes with the lowest SATDs (8 modes for 4×4 and 8×8 blocks and 3 modes for 16×16 and 32×32 blocks) ordered into the RD-list.

The MPM is used after the RMD algorithm. The MPM gets the used modes in the encoded neighbor blocks (the left and above neighbors) and inserts them into the RD-list.

Finally, the RD-cost is fully calculated only for the modes inside the RD-list. The RD-cost calculation generates real data of compression rate and distortion, and then, it is possible to choose the mode of the best performance (lowest RD-cost). These two algorithms (RMD and MPM) reduce a lot the number of RD-cost calculations, significantly decreasing the HEVC intra-prediction complexity.

2.2 Depth modeling mode 1 (DMM-1) algorithm

As previously discussed, the DMM-1 algorithm is based on wedgelets, where a wedgelet is a straight line that segments a block in two regions.

Table 1 clarifies that 3D-HEVC standard has many possible wedgelets in a depth map block. However, 3D-HEVC defines a three-stage search (i.e., main, refinement

and residue) over the complete wedgelet set. Consequently, only a subset of them should be evaluated, resulting in a reduction in the DMM-1 complexity.

Additionally, Table 1 presents the number of wedgelets evaluated before the refinement process and the corresponding decrease, according to the encoding block size. This three-stage search can reduce the evaluation set ranging from 24.7 % to 60.8 %, according to the block size. The complete assessment process signalizes the selected wedgelet and sends the residue between the original depth block and the predicted depth block to the next encoder modules.

Figure 7 presents a high-level diagram of the DMM-1 encoding algorithm, which is composed of three stages: (1) Main Stage, (2) Refinement Stage and (3) Residue Stage. The Main Stage evaluates the initial wedgelet set (i.e., wedgelets that should be assessed before the refinement) and finds the best wedgelet partition among the available ones. The process of finding the best wedgelet requires mapping the encoding block in the binary pattern defined by each wedgelet. According to this mapping, the average values of all samples mapped into regions 0 and 1 are computed, and the predicted block is defined as the average value of each region (*Prediction step*). Next, for each wedgelet pattern, the sum of absolute differences (SADs) is computed using Eq. (1), where $| P_{i,j} - O_{i,j} |$ is the absolute value of the residue between the predicted and original depth samples at position (i, j). Finally, all SADs are compared and the pattern that obtained the lowest SAD defines the best wedgelet (*SAD step*).

$$\mathrm{SAD} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left|P_{i,j} - O_{i,j}\right| \tag{1}$$

The Refinement Stage evaluates up to eight wedgelets around of the selected one in the previous operation (i.e., with a similar pattern). Again, the wedgelet that obtained the lowest SAD among these eight possibilities, along with the first wedgelet selected in Main Stage, is selected as the best one.

Finally, the Residue Stage subtracts the predicted block of the elected wedgelet from the original one and adds this wedgelet into the RD-list.

Figure 8 exemplifies the encoding of a 4 × 4 depth block along with the evaluation of three different wedgelet patterns. Besides, Fig. 8 shows that the prediction process of DMM-1 encodes the depth block sample according to the evaluated wedgelet (i.e., patterns a, b and c). This procedure maps the pixels of the block sample in one of the two regions. Subsequently, the predicted block step computes the average value of all pixels in the region (e.g., the average value of regions 0 and 1 of pattern a is 64 and 76, respectively). The residue step annotates the position corresponding to each pixel with the difference between the predicted and original depth sample. The SAD of all patterns is attained, and finally, the pattern b is elected since it has the lowest SAD.

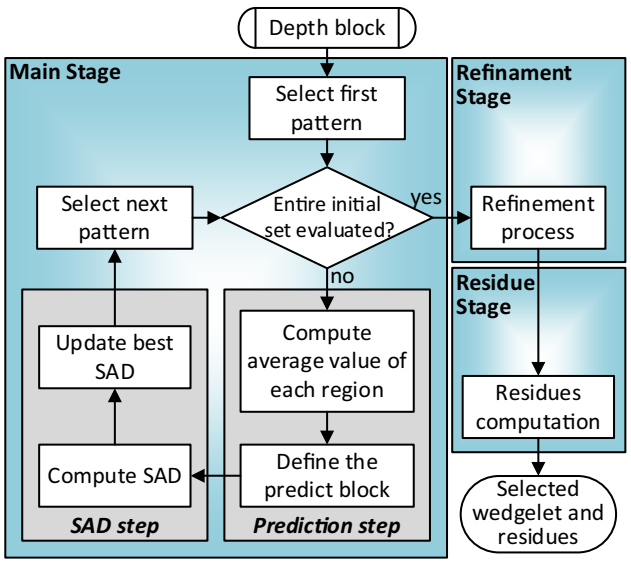

Fig. 7 Main blocks of the DMM-1 encoding algorithm

2.3 Depth modeling mode 4 (DMM-4) algorithm

The DMM-4 algorithm uses a technique called inter-component prediction to find the best contour partition. This inter-component prediction uses previously encoded information from one component (i.e., texture) during the others component prediction (i.e., depth).

The motivation for employing the DMM-4 algorithm is that the encoding depth block represents the same scene than the texture block (previously encoded). Depth data contain considerably different signal characteristics than the texture data; however, it does exhibit some structural similarity to the corresponding texture. For instance, an edge in the depth component usually corresponds to an edge in the texture component [31]. Therefore, there is

Table 1 Number of evaluated wedgelets in DMM-1

Block size	Total possible wedgelets	Evaluated wedgelets	Percentage of reduction
4 × 4	86	58	32.5 %
8 × 8	802	314	60.8 %
≥16 × 16	510	384	24.7 %

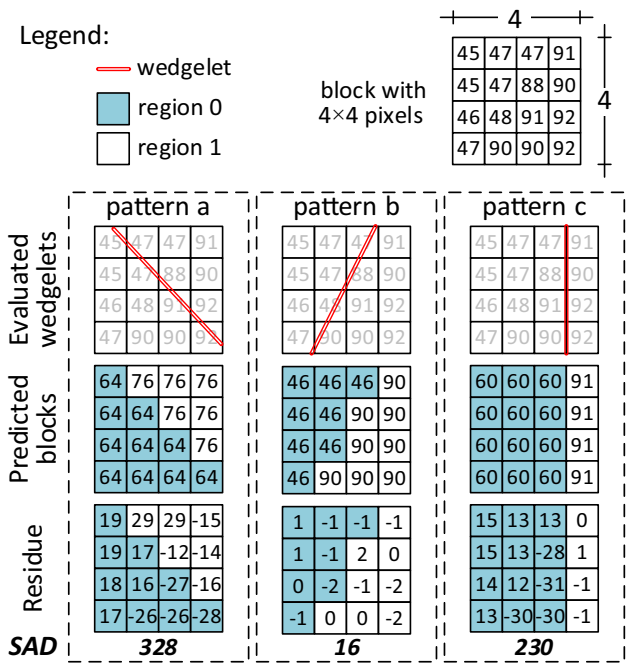

Fig. 8 Example of encoding a block with the DMM-1 algorithm

significant redundant information between both blocks. Figure 9 highlights this redundant information, showing the high similarity and correlation of both blocks. This characteristic motivated the design of DMM-4 by JCT-3V.

Figure 10 presents an example of DMM-4 execution, while Fig. 11 demonstrates the DMM-4 encoding flow. The only information known at the beginning of DMM-4 execution is the texture and depth blocks. Besides, only the reconstructed luminance signal (previously encoded) composes the texture block used as a reference in the DMM-4 algorithm.

Three stages compose the DMM-4 algorithm: Texture Average Stage, Prediction Stage and Residue Stage.

Texture Average Stage starts computing the average value (u) of the texture block using Eq. (2) (143 in Fig. 10), where $T_{i,j}$ denotes the reconstructed texture pixel at (i,j) position.

$$u = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} T_{i,j}}{N^2} \quad (2)$$

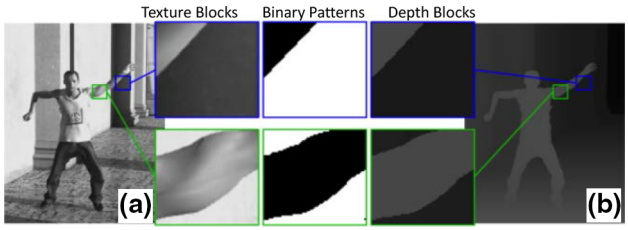

Fig. 9 Correlation between **a** texture and **b** depth components

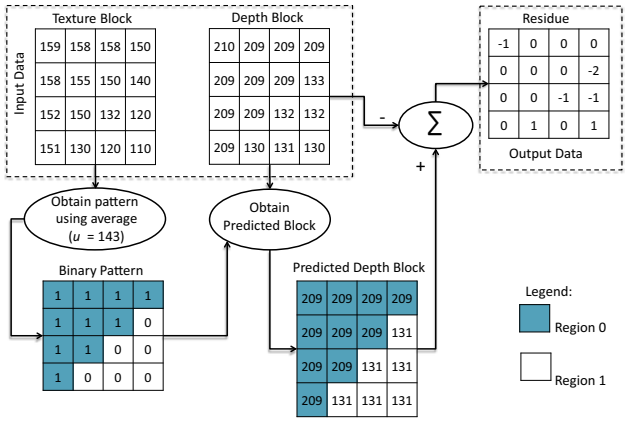

Fig. 10 Example of encoding a block with the DMM-4 algorithm

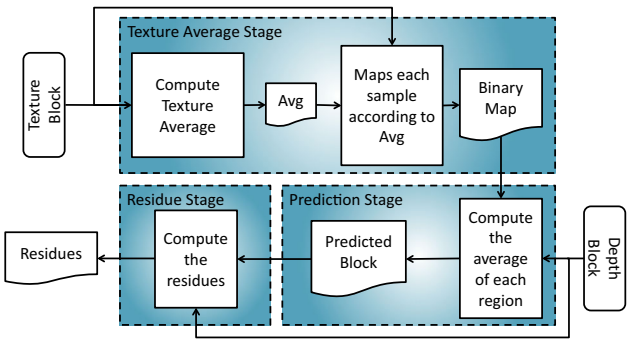

Fig. 11 DMM-4 encoding flow

Let u be an adaptive threshold, the pseudo-code described in Fig. 12 is used to construct a binary map that determines the DMM-4 contour partition. From this pseudo-code, one can conclude that all texture samples smaller than u are mapped to region 0, while all others samples are mapped to region 1, generating a binary map. It is important to notice that it is not necessary depth information during the generation of the binary map. Moreover, considering this information, the DMM-4 pattern does not need to be transmitted because the decoder can use the texture block (which should be decoded before the depth block) and adaptively generate the DMM-4 pattern.

```
1.  for i = 1 to N
2.      for j = 1 to N
3.          if sample[i][j] < u
4.              Region 0 ← sample[i][j]
5.          else
6.              Region 1 ← sample[i][j]
```

Fig. 12 Pseudo-code representing the mapping of samples in regions 0 and 1 in the DMM-4 algorithm

The Prediction Stage processes the depth map using the binary map determined in the previous stage and calculates the average values of all depth block samples mapped in region 0 (AVG_0) and region 1 (AVG_1) (e.g., Fig. 10). These average values are selected as the predicted block, according to the binary map value.

Finally, the Residue Stage of DMM-4 subtracts the predicted block from the original depth block, generating the residues. Next, the RD-cost is then calculated for this DMM-4 block together with the DMM-1 selected block and the blocks defined by RMD and MPM traditional HEVC intra-prediction.

3 Bipartition modes architecture implementation

This section describes our scalable architecture to implement the bipartition modes algorithm. This novel architecture simplifies the DMM-1 algorithm by removing its refinement process (Sect. 4.1 discuss the impacts of this simplification) and maintains the DMM-4 algorithm unaltered.

The Bipartition Core (BP-Core) is the basic structure of the designed architecture, which is replicated according to the size of the encoding block. The architecture has been developed using a regular structure, allowing two levels of scalability: (1) targeting different 3D-HEVC block sizes and (2) targeting various levels of throughput. Section 3.1 presents the BP-Core structure, which can be replicated to reach the scalability required for different block sizes or throughputs. Section 3.2 shows the high-level structure of the proposed scalable architecture.

3.1 Bipartition Core architecture

Figure 13 depicts the BP-Core architecture. The entire bipartition architecture is assembled replicating K times (see Sect. 3.2) an array of $N \times N$ BP-Core in a scalable shape to reach the throughput that supports to encode a $N \times N$ block size.

Each BP-Core receives a pixel value (PIXEL_IN signal) that carries the texture or depth information of the pixel. These pixels are stored in two registers controlled by the ST_TEXT and ST_DEPTH signals, respectively.

Table 2 presents the signals arrangement that controls the BP-Core operation. The MODE signal selects if DMM-1 or DMM-4 should be computed, and the combination of signals OP(0) and OP(1) defines the stage and step of each algorithm.

In our architecture, the implementation of DMM-1 algorithm is composed of three operations: Main Stage, which should perform Prediction Step and SAD Step, and the Residue Stage, which is computed in a single operation:

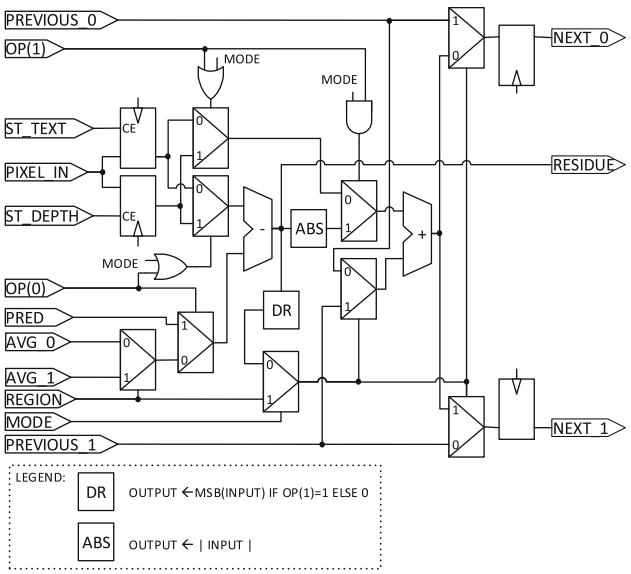

Fig. 13 Schematic of the BP-Core architecture

- Prediction Step (OP(0) = 0 and OP(1) = 0) computes the average depth value of each region. Thus, the architecture selects the depth stored pixel to be added to the previous value of the region that the current pixel belongs (i.e., depending on the signals PREVIOUS_0 or PREVIOUS_1).
- SAD Step (OP(0) = 0 and OP(1) = 1) computes the SAD of each wedgelet pattern. Thus, the stored pixel is subtracted from the average value of the region it belongs (i.e., AVG_0 or AVG_1). The absolute value of this result is added to the values generated by previous BP-Cores (i.e., PREVIOUS_0 and PREVIOUS_1).
- Residue Stage (OP(0) = 1 and OP(1) = 0) computes the residues between the predicted sample and the original signal subtracting PRED, which is the value of the predicted sample, from the stored pixel. The obtained results are sent to the output (RESIDUE).

Table 2 Control signals combination to choose the DMM-1 or DMM-4 algorithm and its execution stage and step

MODE		OP(0)	OP(1)	Stage	Step
DMM-1	1	0	0	Main	Prediction
	1	0	1		SAD
	1	1	0	Residue	–
DMM-4	0	0	0	Texture average	–
	0	0	1	Prediction	–
	0	1	0	Residue	–

The implementation of DMM-4 algorithm is also composed of three operations: Texture Average Stage, Prediction Stage and Residue Stage:

- Texture Average Stage (OP(0) = 0 and OP(1) = 0) computes the average value of the entire texture block. In this step, our implementation of BP-Core sends previous values using NEXT_0 signal (NEXT_1 is only used in latter steps). Similar to the Prediction Step of DMM-1, the architecture selects the stored texture pixel to be added to previous values of PREVIOUS_0 and send this result to the next BP-Core as NEXT_0 signal. In our implementation, mapping each sample into region 0 or region 1 is done in Prediction Stage due to data dependencies.
- Prediction Stage (OP(0) = 0 and OP(1) = 1) subtracts the average value obtained in the previous stage (AVG_0) from the stored texture pixel value. The BP-Core detects if the depth sample should be mapped to region 0 or region 1 according to the most significant bit obtained in the subtraction operation. Besides, the depth stored pixel is added to PREVIOUS_0 or PREVIOUS_1 signal, according to the mapped region.
- Residue Stage (OP(0) = 1 and OP(1) = 0) performs a similar computation than that one carried out in the Residue Stage of the DMM-1 algorithm.

3.2 Bipartition modes architecture

Figure 14 shows a high-level diagram of the proposed architecture for the bipartition modes of the 3D-HEVC. This regular and scalable structure is composed of (1) $K \times N \times N$ BP-Cores array; (2) two memories for storing the DMM-1 patterns (PATTERNS MEMORY) and the division values (DIVIDER MEMORY); (3) four add trees (represented by only one add tree for better visualization) for each $N \times N$ array; (4) two divider circuits (represented by only one divider circuit) for each $N \times N$ array; (5) K comparator circuits; and (6) K register banks. This structure scales to all HEVC block sizes (i.e., 4×4, 8×8, 16×16 and 32×32) expanding the $N \times N$ BP-Core array. Additionally, this structure also enables to vary the data throughput varying the K factor.

The detailed architecture processing flow is described in Fig. 15 for the $1 \times 4 \times 4$ (block size 4×4) case of study, where, as presented in Table 1, 58 wedgelets patterns should be computed in the DMM-1 algorithm along with DMM-4 algorithm. The cycle diagram presented Fig. 15 could be extended to any available block size. Notice that when expanding the architecture by the K factor, the architecture maintains the same cycle diagram by computing K different depth blocks with the same patterns in the same time instant.

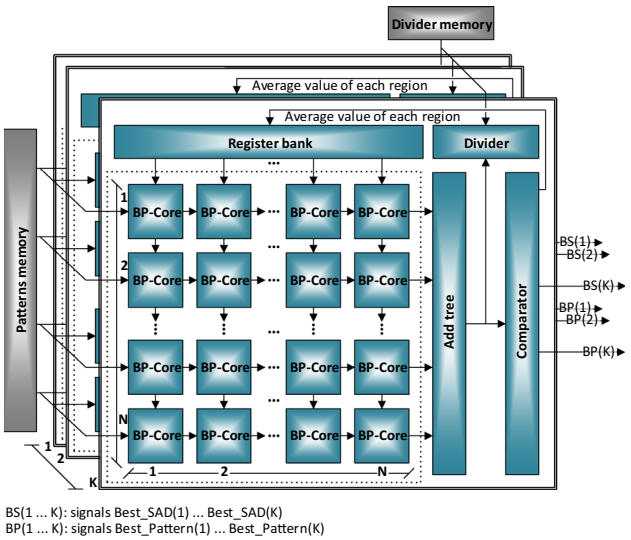

Fig. 14 High-level block diagram of bipartition modes architecture for $K \times N \times N$ blocks

During the first four (N for a generic performance) cycles of execution, 1 (K for a generic performance) array containing four (N for a generic performance) BP-Cores is filled with PIXEL_IN signal (1 byte) containing a line of the depth block. Thus, the generic architecture has a bandwidth of $K \times N$ bytes (4 bytes in this case). As DMM-4 algorithm requires first the texture data to starts it processing and DMM-1 algorithm only requires the depth block, DMM-1 can start being processed, immediately.

As storing pixels does not require passing the data through the processing path, during the next four cycles (N for a generic performance), the texture block is inserted in four (N for a generic performance) BP-Cores per cycle; in the same moment the DMM-1 starts processing the first patterns in its Prediction Step. Moreover, one can notice from Fig. 15 that the DMM-4 algorithm is performed after computing the first six ($N+2$ for the generic case) patterns. The architecture calculates $N+1$ DMM-1 patterns, in pipeline mode with DMM-4 execution, to maximize the architecture performance.

The signals of the BP-Cores positioned in the east column of each $N \times N$ BP-Cores array are added in two add trees (one for region 0 and one for region 1). These results are sent to two dividers in the DMM-1 Prediction Step, which are responsible for obtaining the average value of each region. Besides, the sum of these results is sent to the comparator in the DMM-1 SAD Step, which is responsible for finding the best SAD among all evaluated patterns until that moment. Then, the comparator stores the best SAD, the pattern and the average value of each region.

In the DMM-4 execution, the sum obtained by the add trees is sent to the divider circuit to get the average value of

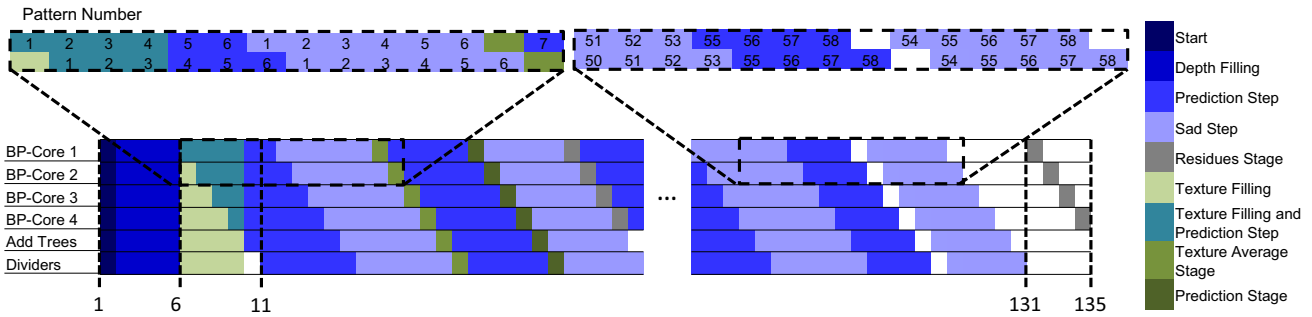

Fig. 15 Cycle diagram of bipartition modes architecture for 1 × 4 × 4 architecture configuration

the texture block in the Texture Average Stage (where only one add tree is used and only one divider is used). Moreover, NEXT_0 and NEXT_1 values of BP-Cores positioned in the east column are added, and its result is divided by the number of elements that composes each region (they are added to the remaining two add trees) during the DMM-4 Prediction Stage.

During both, DMM-1 Prediction Step and SAD Step, and DMM-4 Texture Average Stage and Prediction Stage, the information of region 0 and 1 of the previous BP-Cores (i.e., NEXT_0 and NEXT_1, respectively) is sent in pipeline stages to the next cores. The signals NEXT_0 and NEXT_1 are connected to signals PREVIOUS_0 and PREVIOUS_1 of the following BP-Core to achieve higher performance. The east BP-Cores send their following values to the add tree so that all values are added in one pipeline stage. Moreover, one can notice from the detached patterns in Fig. 15 that the DMM-1 Prediction Step and the SAD step are interlaced.

When a pattern reaches the end of the DMM-1 Prediction Step (i.e., the division is computed), the average value of each region is stored in the register bank, and this information is feedback in the architecture to process the SAD of that pattern.

When all SADs have been computed, the average value of the best pattern is feedback to the register bank according to the region the sample belongs, which will be inserted as the signal PRED in the BP-Cores to compute the residues in N cycles.

In DMM-4, when the total texture average is computed, this value is stored in the register bank, and the information is inserted in the next cycles in the BP-Cores array to allow computing the DMM-4 Prediction Stage. Besides, when this step ends, the average value of each region along with the region is inserted into the register bank, and these values are used to compute the DMM-4 residues in N cycles .

The inclusion of DMM-4 together with DMM-1 algorithm increases only two cycles the time required for an entire block computation when compared to using only the DMM-1 algorithm. Table 3 presents the accurate number of cycles according to the block size regarding only DMM-1 computation and both bipartition modes encoding.

Moreover, the required frequency for real-time encoding 1080p@30fps videos is presented in Table 3, which can be computed using (3), where H and W are the resolution height and width, respectively; $BSize$ is the block size width; $Cycles$ are the required cycles for a given block size (presented in Table 3), and Fr is the frame rate.

Analyzing (3) along with Table 3, one can notice that 8 × 8 blocks require the higher frequency to achieve real-time processing. It happens because, in the first part of the equation, where the number of blocks of a given size inside a frame is computed, there are much more 8 × 8 blocks than 16 × 16 and 32 × 32. Moreover, each of these blocks requires almost the same number of cycles to be calculated; consequently, the 8 × 8 requires higher frequency than the others.

$$\text{Freq (Hz)} = \frac{H * W}{BSize^2} * Cycles(BSize) * F_r \tag{3}$$

4 Experimental results

This section contains the experimental results of this work. Section 4.1 presents an analysis of the impact caused by the DMM-1 refinement removal, and Sect. 4.2 shows the synthesis results for different configurations of the proposed architecture.

4.1 Removal of DMM-1 refinement analysis

The DMM-1 algorithm without the refinement step was evaluated using the HTM-16.0 reference software [25], considering the random access mode and using the videos defined in the common test conditions (CTC) for 3D videos [32]. Table 4 illustrates the results, showing that the removal of DMM-1 refinement step causes a BD-rate [33] degradation of 0.09 % and 0.25 %, in average, for the synthesized views, for random access and all intra cases, respectively.

Table 3 Required clock cycles and frequency according to bipartition mode algorithm and block size

Encoding algorithm	Block size			
	4 × 4	8 × 8	16 × 16	32 × 32
DMM-1 only				
Cycles per block	132	648	816	876
Required Freq (MHz) @30 1080p fps	514	630	199	54
Entire bipartition modes				
Cycles per block	134	650	818	878
Required Freq (MHz) @30 1080p fps	521	631	199	54

Table 4 BD-rate when removing DMM-1 refinement step

Video	BD-rate random access (%)	BD-rate all intra (%)
Balloons	0.16	0.21
Kendo	0.00	0.24
Newspaper_CC	0.19	0.40
GT_Fly	0.02	0.20
Poznan_Hall2	0.03	0.23
Poznan_Street	0.10	0.09
Undo_Dancer	0.08	0.25
Shark	0.15	0.41
Average	0.09 %	0.25 %

Table 5 Memory reduction when DMM-1 refinement is removed

Block size	Memory (bits)		Reduction (%)
	All wedgelets	Without refinement	
4 × 4	1376	928	32.6
8 × 8	51,328	20,096	60.8
16 × 16	130,560	98,304	24.7
32 × 32	130,560	98,304	24.7
Total	313,824	217,632	30.6

Table 5 presents the required memory bits considering that all wedgelets patterns should be stored in a PATTERNS MEMORY inside the architecture. The comparison between storing all wedgelets and removing the refinement shows memory savings higher than 30 %. Due to the large memory resources that should be spent to store all available refinement wedgelets, the removal of DMM-1 refinement in a hardware design presents a sound trade-off between hardware resources, performance and coding efficiency.

In a world dominated by battery-based devices, low-complexity systems are desired. In this direction, our simplification that removes DMM-1 refinement and reduces significantly the hardware complexity is essential for allowing a low-power hardware design.

Simplifications of the DMM-1 wedgelets patterns have already been proposed by other works such as [34], where 27.8 % of reduction is obtained with coding losses of 0.03 % and 0.05 % in random access and all intra. However, such technique requires sharing the memory among different block sizes, which cannot be applicable in the proposed hardware design.

4.2 Hardware results for bipartition modes architecture

We described in VHDL and synthesized for ST 65 nm standard cells technology, the basic configuration of the designed scalable architecture. All synthesis results were intended to achieve real-time processing with HD 1080p@30fps depth maps.

The first four lines of Table 6 present the synthesis results of this experiment targeting different block sizes and using $K = 1$. As one can notice, comparing synthesis frequencies to required frequencies in Table 3, all architecture configurations were capable of reaching real-time processing with HD 1080p@30fps videos.

One can notice that gates and power do not grow linearly when the block size increase. It happens because the number of available wedgelets and the required clock cycles to evaluate an entire block does not grow linearly. Consequently, hardware resources and power consumption have the same behavior, as demonstrated in Table 6.

Our second scalability axis can be used to increase the throughput of the architecture. As in 3D video coding, it can be necessary to encode more than one depth view; our architecture allows increasing its throughput by a factor of K, which multiplies by K the architecture performance. Moreover, when increasing by this K factor, PATTERNS MEMORY and DIVIDER MEMORY should be shared between the different $N \times N$ BP-Core arrays. Consequently, the architecture resources and power consumption would not grow linearly.

As up to the best of our knowledge, this article is the first one that proposes an architecture for all 3D-HEVC bipartition modes; it is not possible to compare with related works. Moreover, the only work that already proposed a bipartition mode architecture was [24], where only FPGA

Table 6 Synthesis results targeting ST 65 nm with comparison against previous works

Target technology	Work	Block size	Area	Registers	Memory bits	Cycles per block	Frequency (MHz)	Latency (cycles)	Processing Rate HD 1080p fps	Power (mW)
ST 65 nm	This	4 × 4	4,070 gates	–	–	134	515.5	134	30.1	22.2
		8 × 8	15,749 gates	–	–	650	632.9	650	30.0	90.9
		16 × 16	49,628 gates	–	–	818	198.8	818	30.0	133.1
		32 × 32	211,950 gates	–	–	878	53.2	878	30.0	166.5
Stratix V 5SGXMABN2F4I3	This	4 × 4	1,389 ALUTs	968	–	134	71.7	134	4.1	–
		8 × 8	4,562 ALUTs	3513	28,672	650	57.8	650	2.7	–
		16 × 16	15,832 ALUTs	15,669	100,352	818	48.8	818	7.4	–
		32 × 32	61,603 ALUTs	80,070	101,238	878	38.1	878	21.4	–
	[24] (DMM-4)	8 × 8 to 32 × 32	5,568 ALUTs	4,405	2,976	74	31.3	115	157	–

results were presented. Considering it, Table 6 also presents synthesis results of the proposed architecture for all available block sizes, targeting an Altera Stratix V FPGA (to allow a comparison using the same FPGA used in our previous work). Furthermore, Table 6 illustrates the results obtained by work [24]. For small block sizes (i.e., 4 × 4 and 8 × 8), the Altera synthesis tool did not include a memory to store the DMM-1 patterns and the divider values because it considered the use of multiplexers instead of memory bits.

The work [24], which is our previous work that implements only the DMM-4 architecture, was not designed to process 4 × 4 blocks because when it was designed, the version of 3D-HEVC reference software did not include 4 × 4 blocks when encoding DMM-4. Also, DMM-4 architecture presented in [24] does not present a regular pattern design and consequently it does not allow easily modifying to insert 4 × 4 blocks or even higher blocks size in futures video coding standards when beyond high definition videos should be used.

5 Conclusions

This work presented a real-time scalable and low-cost architecture for the bipartition modes based on the 3D-HEVC standard, which is composed of both DMM-1 and DMM-4 algorithms. The DMM-1 algorithm was simplified, removing the refinement process and reducing in 30 % the necessary memory to store all available wedgelets (which represents almost 100 Kbits) with a drawback of only 0.09 % of BD-rate in random access mode, in average. The designed architecture merges the execution of DMM-1 and DMM-4 using shared resources to allow better resources usage, smaller power consumption and to achieve higher throughput. The designed architecture was synthesized for standard cell ST 65 nm technology when it achieved real-time processing for HD 1080p videos. It is crucial to detach that the designed architecture is scalable, and it can easily be adapted to process different block sizes and different throughputs according to the application requirements.

References

1. JCT-3V. http://phenix.int-evry.fr/jct2/, access in Ago. 2015
2. Sullivan, G., Ohm, J., Han, W., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. Transactions on circuits and systems for video technology **22**(12), 1649–1668 (2012)
3. Liu, Z., Qiao, Y., Karunakar, A., Lee, B., Fallon, E., Zhang, C., Zhang, S.: H.264/MVC interleaving for real-time multiview video streaming. J. Real-Time Image Proc. **10**(3), 501–511 (2015)
4. Vetro, A., Wiegand, T., Sullivan, G.: Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard. Proc. IEEE **99**(4), 626–642 (2011)
5. Zhang, L., Chen, Y. Karczewicz, M.: Disparity vector based advanced inter-view prediction in 3D-HEVC. IEEE international

symposium on circuits and systems (ISCAS), pp. 1632–1635 (2013)
6. Kang, J., Chen, Y., Zhang, L., Karczewicz, M.: Low complexity neighboring block based disparity vector derivation in 3D-HEVC. IEEE international symposium on circuits and systems (ISCAS), pp. 1921–1924 (2014)
7. Li, X., Zhang, L., Chen, Y.: Advanced residual prediction in 3D-HEVC. IEEE international conference on image processing (ICIP), pp. 1747–1751 (2013)
8. Liu, H., Jung, J., Sung, J., Jia, J., Yea, S.: 3D-CE2.h: results of illumination compensation for inter-view prediction. Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V). Doc. JCT3V-B0045, Shanghai (2012)
9. Kauff, P., Atzpadin, N., Fehn, C., Muller, M., Schreer, O., Smolic, A., Tanger, R.: Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability. Image Commun. **22**(2), 217–234 (2007)
10. Pan, Z., Zhang, Y., Kwong, S.: Fast mode decision based on texture-depth correlation and motion prediction for multiview depth video coding. J. Real-Time Image Process. **11**(1), 27–36 (2013)
11. Zhao, Y., Zhu, C., Chen, Z., Tian, D., Yu, L.: Boundary artifact reduction in view synthesis of 3D video: from perspective of texture-depth alignment. IEEE Trans. Broadcast. **57**(2), 510–522 (2011)
12. Fehn, C.: Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. Stereosc. Disp. Virtual Real. Syst. (SPIE) **5291**, 93–104 (2004)
13. Smolic, A., Muller, K., Dix, K., Merkle, P., Kauff, P., Wiegand, T.: Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems. IEEE international conference on image processing (ICIP), pp. 2448–2451 (2008)
14. Vosters, L., Varekamp, C., Haan, G. Overview of efficient high-quality state-of-the-art depth enhancement methods by thorough design space exploration. J. Real-Time Image Process., pp. 1–21 (2015)
15. Jager, F., Wien, M., Kosse, P.: Model-based intra coding for depth maps in 3D video using a depth lookup table. 3DTV-conference: the true vision-capture, transmission and display of 3D video (3DTV-CON), pp. 1–4 (2012)
16. Shao, F., Lin, W., Jiang, G., Yu, M., Dai, Q.: Depth map coding for view synthesis based on distortion analyses. J. Emerg. Sel. Top. circuits Syst **4**(1), 106–117 (2014)
17. Muller, K., Schwarz, H., Marpe, D., Bartnik, C., Bosse, S., Brust, H., Hinz, T., Lakshman, H., Merkle, P., Rhee, F., Tech, G., Winken, M., Wiegand, T.: 3D high-efficiency video coding for multi-view video and depth data. IEEE Trans. Image Process. **22**(9), 3366–3378 (2013)
18. Tech, G., Chen, Y., Muller, K., Ohm, J., Vetro, A., Wang, Y.: Overview of the multiview and 3D extensions of high efficiency video coding. IEEE Trans. Circuits Syst. Video Technol.(TCSVT) **26**(1), 35–49 (2016)
19. Shen, L., Zhang, Z., An, P.: Fast CU size decision and mode decision algorithm for HEVC intra coding. IEEE Trans. Consum. Electron. **59**(1), 207–213 (2013)
20. Shen, L., Liu, Z., Zhang, X., Zhao, W., Zhang, Z.: An effective CU size decision method for HEVC encoders. IEEE Trans. Multimed. **15**(2), 465–470 (2013)
21. Sanchez, G., Saldanha, M., Balota, G., Zatt, B., Porto, M., Agostini, L.: Complexity reduction for 3D-HEVC depth maps intra-frame prediction using simplified edge detector algorithm. IEEE international conference on image processing (ICIP), pp. 3209–3213 (2014)
22. Lei, J., Li, S., Zhu, C., Sun, M., Hou, C.: Depth coding based on depth-texture motion and structure similarities. IEEE Trans. Circuits Syst. Video Technol. **25**(2), 275–286 (2015)
23. Shen, L., An, P., Zhang, Z., Hu, Q., Chen, Z.: A 3D-HEVC Fast Mode Decision Algorithm for Real-Time Applications. ACM Trans. Multimed. Comput., Commun. Appl. (TOMM) **11**(3), 34:1–34:23 (2015)
24. Sanchez, G., Zatt, B., Porto, M., Agostini, L.: A real-time 5-views HD 1080p architecture for 3D-HEVC depth modeling mode 4. Symposium on integrated circuits and systems (SBCCI), pp. 1–6 (2014)
25. Chen, Y., Tech, G., Wegner, K., Yea, S.: Test model 10 of 3D-HEVC and MV-HEVC. ISO/IEC JTC1/SC29/WG11, Strasbourg (2014)
26. Gu, Z., Zheng, J., Ling, N., Zhang, P.: Fast intra prediction mode selection for intra depth map coding. ISO/IEC JTC1/SC29/WG11, Vienna (2013)
27. Merkle, P., Bartnik, C., Muller, K., Marpe, D., Wiegand, T.: 3D video: depth coding based on inter-component prediction of block partitions. Picture coding symposium (PCS), pp. 149–152 (2012)
28. Merkle, P., Muller, K., Marpe, D., Wiegand, T.: Depth intra coding for 3D video based on geometric primitives. IEEE Trans. Circuits Syst. Video Technol. **26**(3), 570–582 (2015)
29. Zhao, L., Zhang, L., Ma, S., Zhao, D.: Fast mode decision algorithm for intra prediction in HEVC. IEEE visual communications and image processing (VCIP), pp. 1–4 (2011)
30. Cho, S., Kim, M.: Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding. IEEE Trans. Circuits Syst. Video Technol. **23**(9), 1555–1564 (2013)
31. Zou, F., Tian, D., Vetro, A., Sun, H., Au, O.C., Shimizu, S.: View synthesis prediction in the 3-D video coding extensions of AVC and HEVC. IEEE Trans. Circuits Syst. Video Technol. **24**(10), 1696–1708 (2014)
32. Rusanovskyy, D., Muller, K., Vetro, A.: Common test conditions of 3DV core experiments. ISO/IEC JTC1/SC29/WG11 MPEG2011/N12745, Geneva (2013)
33. Bjontegaard, G.: Calculation of average PSNR differences between RD curves. ITU-T SC16/SG16 VCEG-m33, Austin (2001)
34. Shuying, M, Wang, Y., Zhu, C., Lin, Y., Zheng, J.: Reducing wedgelet lookup table size with down-sampling for depth map coding in 3D-HEVC. IEEE international workshop on multimedia signal processing (MMSP), pp. 1–5 (2015)

Gustavo Sanchez is Professor at Federal Institute Farroupilha, Brazil, since 2014. Sanchez received the Electrical Engineer degree from Sul-Rio-Grandense Federal Institute of Education, Science and Technology (2013) and B.S. degree in Computer Science from Federal University of Pelotas (2012). In 2014, he received his M.Sc. degree in Computer Science from Federal University of Pelotas. Sanchez is currently pursuing his Ph.D. degree in Computer Science at Pontifical Catholic University of Rio Grande do Sul. He has 7+ years of research experience on algorithms and hardware architectures for video coding. His research interests include complexity reduction algorithms, hardware-friendly algorithms and dedicated hardware design for 2D and 3D video coding.

César Marcon is Associate Professor at the School of Computer Science of Pontifical Catholic University of Rio Grande do Sul, Brazil, since 1995. He received his Ph.D. in Computer Science from Federal University of Rio Grande do Sul, Brazil, in 2005. His research interests are in the areas of embedded systems on the telecom domain, computer architecture, intra-chip communication architectures, partitioning and mapping application tasks, software & hardware testing, fault tolerance, parallel processing, real-time operating systems. Currently, he participates in five research projects and he works as advisor for MSc. and Ph.D. graduate students.

Luciano Agostini received Ph.D. degrees from the Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, in 2007. He is a Professor since 2002 at the Center of Technological Development (CDTEC) of Federal University of Pelotas, Pelotas, Brazil, where he leads the Group of Architectures and Integrated Circuits (GACI). Since 2013, he is the Research Director of Federal University of Pelotas. He has more than 100 published papers in journals and conference proceedings. His research interests include video coding, arithmetic circuits, FPGA-based design and microelectronics. He is a Senior Member of IEEE and is a member of the IEEE Circuits and System Society, of the IEEE Signal Processing Society, of the Brazilian Computer Society (SBC) and of the Brazilian Microelectronics Society (SBMicro).