doi:10.3772/j.issn.1006-6748.2021.04.012

Wedge template optimization and parallelization of depth map in intra-frame prediction algorithms①

Xie Xiaoyan (谢晓燕)*, Wang Yu②*, Shi Pengfei*, Zhu Yun**, Deng Junyong**, Zhao Huan*
(*School of Computing Science & Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, P. R. China)
(**School of Electronic Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, P. R. China)

Abstract

To reduce the computational complexity and storage cost caused by wedge segmentation algorithm, a scheme of simplifying wedge matching is proposed. It takes advantage of the correlation of the wedge separation line of depth map and the direction of intra-prediction for 3D high-efficiency video coding (3D-HEVC). According to the difference of wedge segmentation between adjacent edge and opposite edge, a set only including 10 4×4 wedgelet templates is given. By expanding of the wedge wave of a certain minimum unit, a simple separation line acquisition method for different size of depth block is put forward. Furthermore, based on the array processor (DPR-CODEC) developed by project team, an efficient parallel scheme of the improved wedge segmentation mode prediction is introduced. By the scheme, prediction unit (PU) size can be changed randomly from 4×4 to 8×8, 16×16, and 32×32, which is more in line with the needs of the HEVC standard. Verified with test sequence in HTM16.1 and the Xilinx virtex-6 field programmable gate array (FPGA) respectively, the experiment results show that the proposed methods save 99.2% of the storage space and 63.94% of the encoding time, the serial/parallel acceleration ratio of each template reaches 1.84 in average. The coding performance, storage and resource consumption are considered for both.

Key words: 3D high-efficiency video coding (3D-HEVC), wedge segmentation, simplified search template, parallelization, depth model mode (DMM)

0 Introduction

As an advance of 2D video, 3D video coding has been drawn in some new characteristics. In order to code for 3D video more efficiently, the multi-view video plus depth (MVD) representation is formed by ISO/IEC MPEG and ITU-T VCEG. Depth information is rendered or synthesized by depth image-based rendering (DIBR)[1]. The increased depth map adds redundancy in processing. Therefore, the wedge segmentation with residual adaptation for depth blocks is presented to reduce the bit rate and increase the quality of rendered view[2]. This strategy has been adopted to extend the high efficiency video coding (HEVC) standard for coding of MVD data[3]. In order to select the optimal template in wedge segmentation, the exhaustive search of wedge-shaped template leads to very high complexity and storage cost of intra prediction.

Recently, experts have been trying various optimization schemes. The main achievement solutions in this field are skipping some templates search for specific blocks[4-7], or making wedge template set simplified[8-9]. Ref.[4] skipped most unnecessary candidates by a simple squared Euclidean distance of variances based rough mode decision, to cope with unacceptable computational burden, based on investigating the statistical characteristics of variance distributions in the two partitions of depth model mode (DMM). Ref.[5] proposed a simplified edge detect algorithm to judge the edge block by extracting the maximum difference between the four corner pixel values of the coding block. DMM mode is executed only for those coding blocks classified as an edge block. To a certain extent, skipping some model searching directly can reduce the coding time, but additional computation overhead for edge detection can not be ignored. Ref.[8] proposed a simplified algorithm which only searches the partition

① Supported by the National Natural Science Foundation of China (No. 61834005, 61772417, 61802304, 61602377, 61874087, 61634004), Shaanxi International Science and Technology Cooperation Program (No. 2018KW-006).
② To whom correspondence should be addressed. E-mail: 1642361376@qq.com
Received on Oct. 26, 2020

patterns in a limited set, based on such a consideration that the separation line in the best matching pattern should be similar to the edge in an actual depth prediction unit (PU). Ref. [9] pointed out that the division direction of the DMM mode should be similar to the prediction direction of the mode in the coarse selection result of the HEVC intra mode, which can be used to reduce unnecessary division calculations in DMM subsection 1 (DMM-1). Those work always try to avoid DMM-1 evaluation in particular conditions. However, none of above can essentially reduce the number of wedge template.

For the sake of real-time processing performance of codec, other researchers attempt to accelerate the DMMs with hardware design. In Ref. [10], five modules composed pipeline architecture was involved in parallel, three of them handle block sizes of 8×8, 16×16, and 32×32, and two identical modules for all 4×4 sizes. The external memorized table stocked the wedgelets and their syntaxes for all block sizes are introduced to simplified calculation. However, the exhaustive approach resulting in high time consumption are not discussed. In Ref. [11], a 5-level pipeline architecture was proposed to accelerate the calculation, in view of the fact that there is no data correlation in wedge segmentation calculation. Although the coding cycle is reduced by 52.3%, but at the cost of 1568 gates are increased. They can only cope with 4×4 depth block. It can not be ignored, there needs to predict 8×8, 16×16, and 32×32 PU size in HEVC. For which, search space and template number are far greater than 4×4. Ref. [12] designed a scalable structure supporting different block sizes, used shared resources to allow efficient area usage, smaller power dissipation, and to achieve higher throughput. It skipped the refinement stage of DMM-1, reserved coarse subnet which still requires a dedicated memory of 217 632 bits to store templates. These parallel schemes although achieve obvious effect in calculation speed, troubles such as different size of depth blocks, large number of templates, and huge search space have not been effectively overcome.

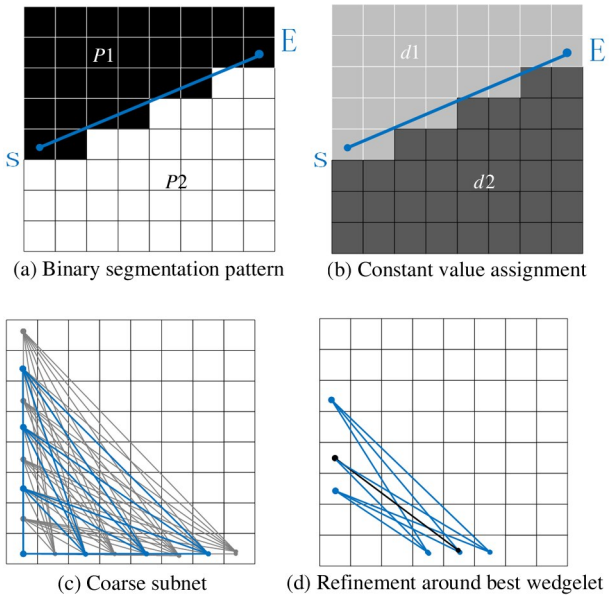
All in all, a practical wedge segmentation prediction needs to consider the realizability and cost in addition of precision. In this paper, a train of thought to simplify the wedge template set on the premise of ensuring quality is proposed. The approach is also motivated by the fact that different from texture map, there often exists large regions with nearly constant values and sharp edges at boundaries in depth map, most of them will choose the simple prediction direction. In the same view, the depth map and texture map depict the same scene. With this kind of characteristics, the direction of the wedge separation line is similar to the prediction direction. The main contribution of the solution lies in three aspects. Utilizing the analysis and statistics to the wedge search template used in depth map prediction of 3D-HEVC, 10 kinds of 4×4 wedge templates with larger prediction probability are screened out. Furthermore, different from represented by piece wise linear functions for each region adopted in DMM-1, a method is proposed to extend the 4×4 wedge template to any $N \times N$ PU, which makes the total number of templates reduced from 3765 to 10. The storage cost and search times of prediction are greatly reduced in turn. Based on the computing pattern of the array processor (DPR-CODEC) developed by project team, an efficient paralleling scheme is also detailed to optimization method mentioned above. The proposed simplified search template and parallel DMM-1 scheme are verified with test sequence in HTM16.1. The experiment results show that the proposed methods save 99.2% of the storage space and 63.94% of the encoding time, the average serial/parallel acceleration ratio of each template is 1.84.

The remainder of this paper is organized as follows. The principle of wedge segmentation is introduced in Section 1. The details about templates simplifying rules and improving idea of DMM-1 are given in Section 2. The mentality of extending the 4×4 wedge template to any $N \times N$ PU and parallelization are also analysed. In Section 3, the improved DMM-1 algorithm is parallelly designed. Experiments are conducted in Section 4. Finally, Section 5 draws some concluding remarks.

1 Wedge segmentation algorithm

As detailed in Refs[2,3], the wedgelet model of a depth block is obtained by partitioning the area of the block into two segments, labeled by $P1$ and $P2$ in Fig. 1(a), separated by a straight line. For a depth block of size $N \times N$, the separation line is defined by the sample positions of start point S and end point E. From this, the segmentation pattern is represented as an array of binary elements with size $N \times N$, namely partition information table, which determines whether the corresponding sample belongs to segment $P1$ or $P2$. According to the segmentation pattern, the constant depth values $d1$ and $d2$, namely constant partition value (CPV), are assigned to samples belonging to $P1$ and $P2$ respectively. The wedgelet model of a block is generated, as shown in Fig. 1(b). Deriving the optimal wedgelet model for a given depth block means carrying out a minimum distortion search for all possible wedge

segmentations. This includes two steps, coarse subnet and refinement around best wedgelet. In the first step, the minimum distortion wedgelet is searched for a coarse subset of segmentations, with a certain interval in all possible positions, shown as Fig. 1(c). In the second step, the segmentation is refined by searching with 8 adjacent precisions around the best coarse wedgelet as the example shown in Fig. 1(d).



(a) Binary segmentation pattern (b) Constant value assignment

(c) Coarse subnet (d) Refinement around best wedgelet

Fig. 1 Wedge segmentation of a depth block

2 Improvement of DMM-1

2.1 Simplified wedge templates set

According to the start and end position of separation line, the wedgelet models can be differentiated into two major categories, the adjacent edge partitions and the opposite edge partitions. For adjacent edge partitions, the starting and ending positions of separation line belong to adjacent vertical and horizontal boundary lines of the current PU, indicated with a *wedgeOri* ranging from 0 to 3 as shown in Fig. 2. While opposite edge partitions have start position and end position belonging to parallel boundary lines of the current PU, indicated with a *wedgeOri* equal to 4 or 5 as shown in Fig. 2. The line end position offset for refining the wedgelet partition can't be predicted, but sought by iterating over a range of offset values and comparing the distortion of the different results.

In addition, the depth map coding also uses the quadtree coding tree unit (CTU) recursion similarly consistent with HEVC. Four types of block size need to be segmented. But the resolution step size used for generating the wedgelet patterns depends on the block size. For 16×16 and 32×32 blocks, the possible start and end positions are restricted to locations with 2-sample accuracy. For 8×8 blocks, full-sample accuracy is used, and for 4×4 blocks, half-sample accuracy is used. From this, the number of coded block wedge templates of different sizes is different. The larger the size, the larger the number of corresponding templates. The exhaustive searches for the optimal wedge template from tens of thousands of templates bring about a huge amount of computation. At the same time the huge storage consumption is also a problem, as shown in Table 1.

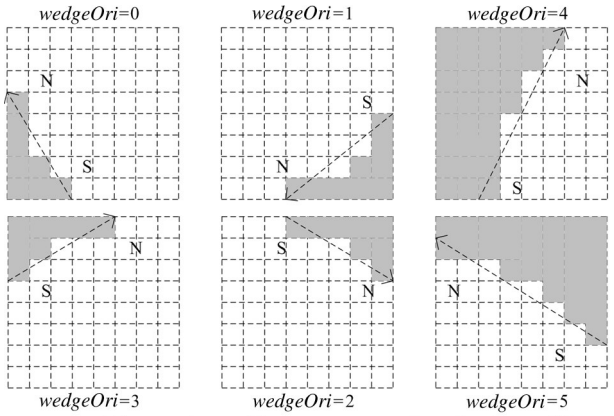


Fig. 2 Wedgelets partition selection

Table 1 Number of wedgelet patterns for different PU sizes

PU size	Number of wedgelet patterns[4]	Bits of one pattern	Total size /bits
4×4	86	16	1376
8×8	766	64	49 024
16×16	510	256	130 560
32×32	510	1024	522 240

Although DMM-1 improves the accuracy of intra prediction greatly, the new wedge segmentation also brings about greater computational complexity and storage consumption. All these designs are tolerable for software encoder, just a matter of coding performance. But it is disastrous for hardware implementation, due the limitation of resource conditions and fixed functional circuit on chip. This paper is to make the algorithm suitable for hardware, and not deviate from the original intention of original designer. Therefore, it is necessary to find a more simply way to acquire wedgelet and fewer templates. To this end, the simulation test is performed with various types of video test sequences, under full DMM-1 in HTM16.1. The results of statistical analysis are shown as Table 2. Six optimal candidate directions, horizontal mode, vertical mode, Ang-5, Ang-14, Ang-2, and Ang-30, can be obtained. The average probability of those reaches 80.19%. The

average probability of rest 27 directions is less than 20%. This is the fact that there often exists large regions with almost constant and stationary samples, and sharp edges at boundaries in depth map. These regions often select the simple directions in intra-prediction.

Table 2 Statistical probabilities of the best prediction direction

Sequences	6 candidate directions/%	Other directions/%
$QP=27$	67.21	32.19
$QP=32$	78.21	21.97
$QP=37$	85.39	14.61
$QP=42$	89.95	10.05
Average	80.19	19.81

Referring to the conclusions in Ref. [8] and Ref. [9], based on 6 directions mentioned above and 6 types of wedge pattern in Fig. 2, a wedge template set consisting of 10 4×4 templates is derived, shown as Fig. 3. The horizontal and vertical directions only apply to opposite edge, by the centre line. The other 4 directions are applied to adjacent edge and opposite edge separately, employ angles defined in their candidate patterns. Verified by HTM16.1, this template set can work well.

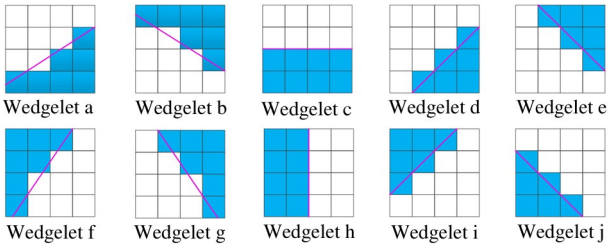


Fig. 3 Wedge template set

2.2 Template extension for $N\times N$ PU

After analysis of the results of simulation test, it can be found that the matched best wedge block always appears in middle position of the coding depth. In view of the correlation between adjacent pixels, it is possible to generate a separation line from the basic 4×4 template, and skip the over-complicated optimal template search for the $N\times N$ PU. Take a wedge template of 8×8 PU as an example, shown as Fig. 4.

The dash line is the wedge separation line. Its function is derived by the starting position (X_1, Y_1) and the ending position (X_2, Y_2) of the wedge separation line of the middle basic wedge template. That is a linear function, $Y = ax + b$. a and b are calculated according to Eq. (1).

$$\begin{aligned} a &= (Y_2 - Y_1)/(X_2 - X_1) \\ b &= (X_2Y_1 - Y_2X_1)/(X_2 - X_1) \end{aligned} \tag{1}$$

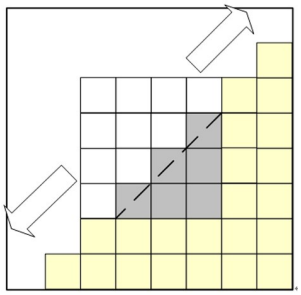


Fig. 4 Sketch of the wedge template extension

On the basis of this separation line, the value $P_{i,j}$ in binary mapping matrix is assigned according to Eq. (2).

$$P_{i,j} = \begin{cases} 1 & j(X_2 - X_1) - i(Y_2 - Y_1) < (X_2Y_1 - Y_2X_1) \\ 0 & j(X_2 - X_1) - i(Y_2 - Y_1) \geqslant (X_2Y_1 - Y_2X_1) \end{cases} \tag{2}$$

2.3 Minimum template set based wedge segmentation prediction

According to above discussion, wedge segmentation prediction for any PU size only uses ten 4×4 templates given in Fig. 3, which makes the total number of templates reduced from 3765 to 10. And total size of searching table is reduced from 1 947 360 to 160 bits. Combined with other steps detailed in DMM-1[13], the flow chart of new algorithm is given in Fig. 5, 8 steps are discussed in detail as follows.

Step 1 Initialisation. Initialize the variables, which store the best matching template and the calculation results based on it, including the best template identifier N and its corresponding residual matrix $\boldsymbol{A}_{residual}$, *BestSAD*, *BestCPVs*. In the proposed algorithm, the coarse subnet and refinement for optimal template search is replaced with the most suitable template matching and extension in wedge segmentation procedure. So there only needs to deliver the identifier, instead of the binary mapping matrix of the best matching template.

Step 2 Edge block checking. Load a PU by using corner detection operator to detect whether the current PU contains edge information with method in Ref. [13]. If it is a flat block, the traditional HEVC intra prediction is applied. Otherwise, repeat Steps 3 to 7 for 10 templates in Fig. 3.

Step 3 Wedge separation line extracting. Select one of the templates from the wedge template set. Extract the starting position (X_1, Y_1) and the ending position (X_2, Y_2) of its wedge separation line.

Step 4 Template extension. Judge whether the current PU size is 4×4. If it isn't, using the method detailed in subsection 3.2 to extend the current tem-

plate to PU size. Otherwise, use the current template directly.

Step 5 CPVs calculating and reference block construction. Generating the binary mapping matrix according to the template, referring to Eq. (2). Calculating the CPVs according to it and the pixels of PU. Constructing the reference block matrix through CPVs filling.

Step 6 Residual matrix and sum of absolute difference (SAD) value calculating. Calculate the residual matrix by pixels in PU and reference block. The SAD is used to evaluate the distortion predicted by the current template in this paper. It is calculated according to Eq. (3), where, k is the template identifier, $PU(i, j)$ is the pixel value of the i row and j column in the depth block PU, $g_k(i, j)$ is the value of the i row and j column in the reference block matrix, N is the size of PU. The smaller the SAD value, the lower the distortion.

$$SAD = \sum_{i=1}^{N} \sum_{j=1}^{N} | PU(i, j) - g_k(i, j) | \tag{3}$$

Step 7 The best prediction selection. Compare the current SAD value with the *BestSAD*, if it is smaller, record template identifier and its corresponding residual matrix $\boldsymbol{A}_{residual}$, *BestSAD*, *BestCPVs* by this calculation to the best matching variables. Otherwise, keep the best matching variables unchanged.

Step 8 Output the best prediction results.

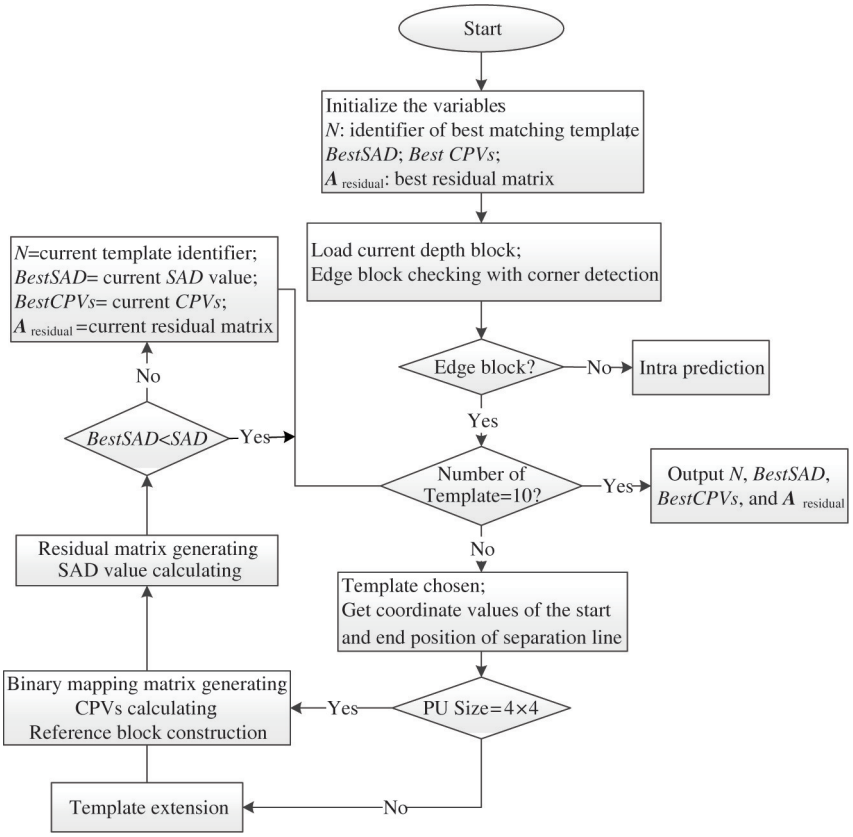


Fig. 5 Flowchart of minimum template set based wedge segmentation prediction algorithm

3 Parallelization

3.1 Hardware architecture

As shown in Fig. 5, the coarse subnet and refinement for optimal template search are replaced with the most suitable template matching and extension in wedge segmentation procedure. At the same time, the storage cost and search times are saved significantly for the new simplified strategy. It is possible to implement all PU size wedge segmentation on resource limited field programmable gate array (FPGA). Furthermore, only the original pixel value of PU and current wedge template are involved in during the template matching. The wedge segmentation operation and data calculations of different templates are no correlation. It means that the prediction process of different templates can be carried out in parallel for a depth PU. Furthermore, the prediction data for each template can be loaded at the same time, which can reduce the pretreatment time by 2/3. In this section, the improved DMM-1 for parallel

processing is optimized by the dynamically programmable reconfigurable CODEC (DPR-CODEC), which is a dynamic reconfigurable array processor to cope with the special demand of HEVC.

The DPR-CODEC is composed of 1024 PEs in the form of adjacent interconnection. Only 4 × 4 PEs are assigned to the DMM-1 for this paper, clearly demonstrated in Fig. 6. The size of each PE instruction and data sharing storage can be adjusted dynamically. Data input memory (DIM) is a data buffer for caching coding block. Data output memory (DOM) is a data buffer for caching reference block and reconstructed image. The global controller decides the operating mode and chooses the appropriate functions for the PE or PEs. Each PE contains 16 registers, including 12 local registers and 4 shared registers. The shared registers distribute in east, south, west, and north directions, named RE, RW, RS, and RN respectively. PE can access each other through shared register. There are two ways of data interaction between PEs. Mode 1 is shown in the dashed arrow, and mode 2 is shown in the solid arrow. Mode 1 is used to send data from the local registers (R3, R4, R5 and R6) directly to the execution units of the neighboring PE as the source operations. Mode 2 can transfer data from local PE to adjacent registers through shared register, and the data can be operated in the subsequent processing. One of the data interaction method is using the adjacency interconnection structure. Another is the distributed shared storage structure under unified addressing mode, it can also be realized through the high-speed switching unit.

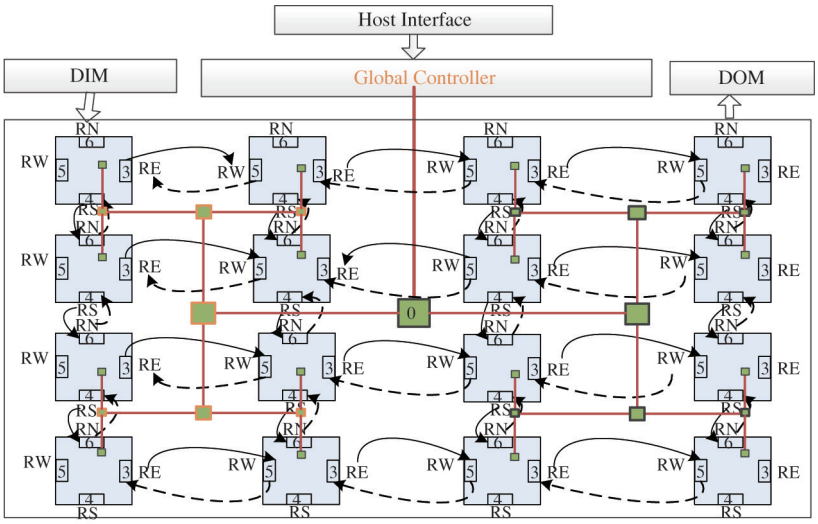


Fig. 6 Hardware architecture for proposed algorithm

3.2 Parallel design scheme

Based on DPR-CODEC, an extending template can be performed in such manner. The basic template is stored and transmitted in binary mapping matrix. When the size of PU is larger than that of template, lock the position of the template in PU by $(\frac{N}{2}-2,\frac{N}{2}-2)$ in the first, where N is the PU size. So, the coordinates of the four corners of the template are $(\frac{N}{2}-2,\frac{N}{2}-2)$, $(\frac{N}{2}-2,\frac{N}{2}+2)$, $(\frac{N}{2}+2,\frac{N}{2}-2)$ and $(\frac{N}{2}+2,\frac{N}{2}+2)$, respectively. For an 8 × 8 PU, the position is (2, 2), the coordinates are (2, 2), (2, 6), (6, 2), and (6, 6), respectively. Then fill positions within the four corners with template. The other positions are filled according to Eq. (2). The original CPVs coding manner is adopted in this paper. Because of the complexity of the predicted and delta CPVs, neighboring blocks pixels need to be kept for the CPV predictors. Those are changing along with the template, which makes more storage resource and control logic in hardware realization. Besides the original CPVs bring about the best approximation for the given partition.

The function allocation of the proposed algorithm on DPR-CODEC is shown in Fig. 7. The depth map is loaded into DIM in advance. PE00 is used for depth PU loaded and transmitted to other PEs. PE01 is for edge block checking. PE03 is for SAD comparison and output the results. PE10 to PE33 are assigned to perform Step 3 to Step 6 in parallel mode, detailed in the algorithm designed in sub section 2.3.

In DPR-CODEC, the dynamically programmable and reconfigurable mechanism can issue different in-

structions and data to each PE through globe controller and H-tree network. The adjacent interconnection and data sharing storage make data access among PEs within one hop but cascade, which makes different data loading to each PE faster and much easier. Although filling modes of each template is different, it can be operated by different PE at the same time. Because the template for every PU size is same in the proposed algorithm, the PU size can be changed randomly from 4 × 4, 8 × 8, 16 × 16, to 32 × 32, which is more in line with the needs of the HEVC standard. The specific is introduced by taking a 8 × 8 PU as an example.

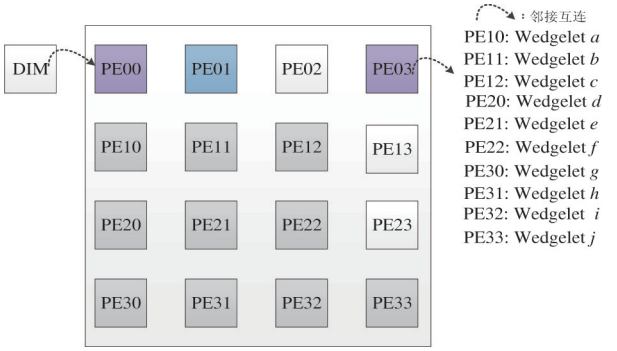


Fig. 7 Mapping flow chart of proposed algorithm

Step 1 Data loading. PE00 accesses the DIM through the adjacent interconnect mode in register R10, reads the corresponding pixels of current PU.

Step 2 Data distribution. PE01, PE10, PE20 and PE30 read original pixels from PE00 via shared registers at the same time, controlled by handshakes. As soon as they receive the data, each PE forwards the data to the appropriate PE in the same way. The order of data spreading is from PE10 to PE11, and PE12, from PE20 to PE21, and PE22, from PE30 to PE31, PE32 and PE33. PE01 does edge detection instead of spreading. If edge detection result is flat then trigger intra-prediction handshake, or else wedge matching.

Step 3 Parallel template expansion. If it is needed, PE 10 to PE 33 start the template expansion after getting the pixels data, according to the data loading condition. The block size had been judged out while loading. Ten templates had been loaded to each PE while arraying initial configuration, which benefits from fixed template.

Step 4 CPV sencoding and SAD solving. PE10 to PE33 perform Step 3 to Step 6 in parallel mode, detailed in the algorithm designed in Section 2.3, obtaining their corresponding $\boldsymbol{A}_{residual}$, *SAD*, and *CPVs*.

Step 5 The best prediction result obtaining and output. The predicted results calculated by each PE are sequentially sent to PE03. PE03 obtains the optimal template through comparing each SAD, reads its corresponding identifier, $\boldsymbol{A}_{\text{residual}}$, *SAD*, and *CPVs* via shared registers, and outputs them.

These are achievable by the dynamically reconfigurable mechanism. DPR-CODEC is a dynamic reconfigurable array processor, it can change instructions at runtime. The instructions in PEs are issued and stored while initial configuration. The global controller decides the operating modes, and distributes different call instructions for the PE or PEs.

4 Experiment results and discussion

This section presents the results for the proposed wedge segmentation DMM scheme. To evaluate the feasibility of the proposed algorithm, software simulation results present the coding quality and time consumer. Several sequences recommended by the JCT-3V group are encoded, including Kendo, Newspaper1, and Balloons with resolution of 1024 × 768, as well as Undo _ Dancer, Poznan _ Hall2, and Poznan _ Street (1920 × 1088), which are tested under the common test conditions (CTC) specified in Ref. [14]. In order to verify the performance of the parallel scheme, verification method is used as follows. Firstly, modify the configuration file of the test model HTM16.1, obtain test data and block partition information, and store it in the off-chip memory. And then, use QuestaSim to map the reconfigurable scheme to the dynamic reconfigurable array. The instructions of the parallel program are initialized to instruction memory. The structure of the reconfigurable array is verified by QuestaSim-64 10.1.d. After functional simulation, the design is synthesized through the Xilinx virtex-6 FPGA with BEE4 XC6VLX365T-1FF1156 with speed grade-1.

4.1 Effect of improved algorithm

Since the proposed techniques focus on wedge segmentation in depth intra coding, all test sequences are encoded using the intra-only structure. The objective evaluation of availability is calculated by the peak signal-to-noise ratio (PSNR) loss, and depth coding time saving ΔT. ΔT represents depth coding time change compared with the benchmarking algorithms, which is defined as

$$\Delta T = \frac{T_{proposed} - T_{htm.\ original}}{T_{htm.\ original}} \times 100\% \tag{4}$$

where, $T_{htm.\ original}$ is the intra-only depth coding time in HTM16.1 software, $T_{proposed}$ is the depth coding time of the proposed DMM-1 algorithm. Positive and negative values denote increments and decrements, respectively. The simulations are conducted on a 64 bits MS

Windows 10 OS running on an Intel(R) Core(TM) i7-8565U CPU of 1.80 GHz and 8.0 GB RAM.

To reveal the quality of the proposed algorithm, 7 groups of video sequences with different moving intensity and resolution are evaluated. The simulations are carried on Matlab with 50 frames for each kind of video, and the results are shown in Fig. 8. It is possible to note that the PSNR loss is 7.31 dB in average, that is slightly higher. Because the planner is applied only for flat block after corner detection to shield the interference of intra-prediction parts, which is much lower on whole intra-prediction of depth map. But that is not the focus of this paper. So, PSNR loss is up to 10.55 dB for Poznan _ Hall2 with more flat blocks. However, the PSNR loss is down to 3.78 dB for GT _ fly which has fewer flat blocks. These results are expressive for the coding quality of proposed algorithm, which is acceptable.

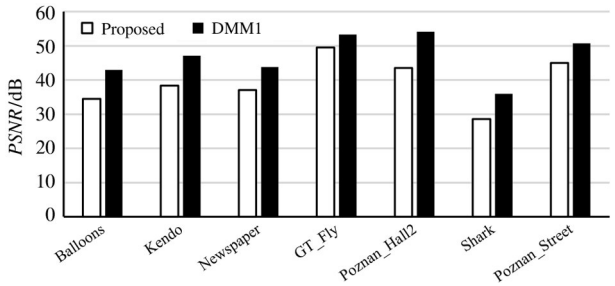


Fig. 8 PSNR of proposed algorithm for CTC evaluation compared with DMM-1 benchmarking

From Table 3, it is obvious to note that the improvement achieves a depth coding time saving of 63.94 % in average. Compared with Ref. [4] and Ref. [5], the optimization effect is more significant, whether it is 1024 × 768 or 1920 × 1088 resolution. The improvement measures in Ref. [4] are mainly concentrated on simplifying the rate distortion cost (RD-cost) computational complexity of intra-mode selection for flat. That is why it is more effective for gentle sequence, such as Kendo and Poznan _ Hall2, compared with violent changes form, like Balloons and Poznan _ Street. In Ref. [5], it only skipped DMM-1 process for flat block, number of mode selection and calculation method are not involved. Its time saving effect is less obvious. This paper focuses on flat and edge block at the same time. The corner detection is used to judge the type of block, and directly apply the planar pattern for flat one based on the texture characteristic of depth map. In addition, there are only 10 wedge templates for the edge block, the template selection process is greatly simplified. That is why the time saving is stable around 64% whether it is edge or not.

Table 3 Depth coding time comparison under CTC

Test sequence \ Δt/%	Ref. [4]	Ref. [5]	This work
Kendo	−38.23	−11.1	−60.76
Newspaper1	−33.85	−12.8	−63.82
Balloons	−35.49	−10.3	−67.96
1024 × 768	**−35.86**	**−10.7**	**−64.18**
Undo _ Dancer	−41.57	−10.3	−62.36
Poznan _ Hall2	−48.33	−12.5	−65.33
Poznan _ Street	−39.67	−11.7	−63.41
1920 × 1088	**−43.19**	**−11.00**	**−63.70**
Average	−39.52	−11.4	−63.94

4.2 Performance of parallelization

Based on BEE Cube's BEE4 hardware experimental platform, the team paralleled the improved DMM-1 based on DPR-CODEC array processor, synthesized with Xilinx ISE14.7. In order to improve the efficiency of DMM-1, the H-tree network and adjacency interconnection structure are fully utilized in data loading. It makes video data loading parallelized and save the data loaded time. As shown in Table 4, it saves about 50% time on data loading. The parallel data loading speed-up ratio reaches 2.169 compared with sequential manner.

Table 4 Sequential/parallel data load time comparison (unit: clock cycle)

Block size \ Loading mode	Sequential loading	Parallel loading	Speed-up ratio
4 × 4	2955	1369	2.159
8 × 8	8587	3927	2.187
16 × 16	28 299	13 729	2.061
Average			2.169

(Note: the above table contains the current coding block and the corresponding reference pixels)

The complexity of each wedge-shaped templates is different, time consumption is also different. Table 5 accesses the coding time for 8 different templates. The parallel coding time elapsed from the beginning of parallel computing to the time when the last processor produces output results. The serial encoding time is collected from serial algorithm performed on a single PE. The speed-up ratio $sp(n)$ is used to evaluate the degree of improvement in running time caused by the parallelism of the algorithm, which is defined as

$$sp(n) = ts(n)/tp(n) \tag{5}$$

where, $ts(n)$ is the running time of the fastest serial algorithm, $tp(n)$ is the running time of parallel solving the same problem. As shown in Table 5, the speed-up ratio of paralleled implementation of wedge segmenta-

tion proposed in this paper is 1.84 in average.

Let p denote the number of processors. When $sp(n) = p$, it is called linear acceleration. Take 16 × 16 as an example. In the experiment, $sp(n) = 2.103$, $p = 12$, $sp(n) \leqslant p$ is caused by the difficulty of decomposing a computing problem into some parallel sub-problems, or by the fact that too much communication is involved in the computation of various sub-problems. In terms of the number of processors, cost $c(n)$ is the total number of execution steps when solving a problem, which is the product of run time $t(n)$ and number of processors $p(n)$, which is defined as

$$c(n) = t(n) \cdot p(n) \tag{6}$$

Efficiency $Ep(n)$ reflects the utilization of processors in parallel systems, which is defined as

$$Ep(n) = sp(n)/p(n) \tag{7}$$

where $p(n)$ denotes the function of the processor with respect to the problem size n. $Ep(n) = 0.159$ in the experiment accords with the theoretical value of $0 < Ep(n) = 1$. The practicability and effectiveness of the parallel scheme are fully verified.

In Table 6, the comparison of the synthesizing results with Ref. [10], Ref. [11] and Ref. [12] is given. For this work, it only spends 210K equivalent gates. The hardware resource cut down is apparent. That benefits from number of wedgelets reducing. Moreover, it only performs 23 cycles for 4 × 4 PU, because no template extension operation is involved. But the disadvantage of execution time exists in 16 × 16 PU,

Table 5 Parallel template performance assessment (unit: clock cycle)

Wedgelets in Fig. 3	8 × 8			16 × 16			32 × 32		
	Serial encoding time	Parallel coding time	Speed-up ratio	Serial encoding time	Parallel coding time	Speed-up ratio	Serial encoding time	Parallel coding time	Speed-up ratio
Wedgelet a	10 287	6019	1.709	39 978	17 680	2.261	150 097	78 291	1.917
Wedgelet b	10 292	6121	1.681	37 381	16 654	2.245	121 988	69 123	1.765
Wedgelet c	11 281	6986	1.645	36 964	18 396	2.009	163 019	98 257	1.660
Wedgelet d	9320	4829	1.930	37 207	18 721	1.987	114 632	82 362	1.392
Wedgelet e	9320	4569	2.040	42 381	19 981	2.121	102 903	78 672	1.308
Wedgelet f	11 288	7219	1.564	39 957	21 314	1.875	142 236	81 477	1.746
Wedgelet g	11 296	7033	1.606	37 496	15 956	2.350	183 354	104 920	1.748
Wedgelet h	11 271	6123	1.841	37 473	18 937	1.979	174 231	97 317	1.790
Average	10 316	6112	1.752	38 605	18 455	2.103	144 020	86 427	1.666

(Note: The above table contains the current coding block and the corresponding reference pixels)

Table 6 Comparison of the synthesizing results with related work

Work	Technology	PU Size	Memory for wedgelets store/bits	Execution time/cycles	△M/%	Area
This work	Xilinx FPGA Virtex 6	4 × 4	160	23	99.2%	210K equivalent gates
		8 × 8	0	100		
		16 × 16	0	1272		
		32 × 32	0	25 355		
Ref. [10]	Xilinx FPGA Virtex 6	4 × 4	1376	17	0%	330K equivalent gates
		8 × 8	50 048	21		
		16 × 16	356 864	35		
		32 × 32	1 539 072	119		
Ref. [11]	Nangate 45	4 × 4	1376	63	——	4895K gates
Ref. [12]	ST28nm	4 × 4	928	134	30.6%	88K gates
		8 × 8	20 096	650		
		16 × 16	98 304	818		
		32 × 32	98 304	878		

especially in 32 × 32, for template extension operation. This is to offset the cost of storage space. There only needs 160 bits to all wedgelets, saving reaches 99.2%. Although Ref. [10] retrenched some wedgelets by skipped refinement stage and resulted in a lower gate count, it still requires a dedicated memory of 217 632 bits to store coarse templates. Ref. [11] paralleled DMM-1 by 5 levels pipeline architecture, decreasing the PU handle time at the costs of 4895 gates. And it only treats PU, unable to cope with flexible PU size. Ref. [12] treated 32 × 32 PU at only 119 cycles through storing all wedge templates form lookup table. So, it requires a dedicated memory of 1 947 360 bits, and its hardware resource consumption is 2.54 times of the method proposed in this paper.

5 Conclusion

Aiming at the huge data processing problem in the process of 3D-video coding, a simple wedge segmentation prediction method is proposed, considering the realizability and cost. The main contribution of the solution is three aspects. Utilizing the analysis and statistics to the wedge search template used in depth map prediction of 3D-HEVC, 10 kinds of 4 × 4 wedge templates are screened out with larger prediction probability. Furthermore, a method is proposed to extend the 4 × 4 wedge template to any $N \times N$ PU, which makes the total number of templates reduced from 3765 to 10. The storage cost and search time are reduced. Based on the computing pattern of the array processor (DPRCODEC) developed by project team, an efficient paralleling scheme is also detailed. The proposed simplified search template and parallel DMM-1 scheme are verified with HTM16.1 under CTC. The experiment results show that, the proposed methods save 99.2% of the storage space and 63.94% of the encoding time, the data loading time saves about 50%, and serial/parallel acceleration ratio of each template is 1.84 in average. It not only guarantees the coding quality, but also solves the complexity in the encoding process and improves the operation efficiency. In the future, the coding quality enhancing by optimizing intra process of flat block will be considered.

References

[1] MPEG Video Group. Applications and Requirements on 3D Video Coding, ISO/IECJTC1/SC29/WG11[R]. Geneva: MPEG Video Group, 2011

[2] Domanski M, Konieczny J, Kurc M, et al. 3D video compression by coding of disoccluded regions[C]// The 19th IEEE International Conference on Image Processing, Orlando, USA, 2013:1317-1320

[3] Karsten M, Heiko S, Detlev M, et al. 3D high-efficiency video coding for multi-view video and depth data[J]. *IEEE Transactions on Image Processing*, 2013, 22(9): 3366-3378

[4] Zhang H, Fu C, Chan Y, et al. Probability-based depth intra-mode skipping strategy and novel VSO metric for DMM decision in 3D-HEVC[J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018, 28(2): 513-527

[5] Saldanha M, Sanchez G, Zatt B, et al. Energy-aware scheme for the 3D-HEVC depth maps prediction[J]. *Journal of Real Time Image Processing*, 2017, 13(1):1-15

[6] Park C. Edge-based intramode selection for depth-map coding in 3D-HEVC[J]. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, 2015, 24(1):155-162

[7] Liu P. Research on the fast selection method of depth map intra prediction DMM mode in 3D-HEVC[D]. Xi'an: School of Communication Engineering, Xidian University, 2017: 27-44 (In Chinese)

[8] Lei J, Sun Z, Go Z, et al. Simplified search algorithm for explicit wedgelet signalization mode in 3D-HEVC[C]// IEEE International Conference on Multimedia and Expo, Hong Kong, China, 2017: 805-810

[9] Zhang M, Zhao C, Xu J, et al. A fast depth-map wedgelet partitioning scheme for intra prediction in 3D video coding[C]//2013 IEEE International Symposium on Circuits and Systems, Beijing, China, 2013: 2852-2855

[10] Amish F, Bourennane E. An efficient hardware solution for 3D-HEVC intra-prediction[J]. *Journal of Real-Time Image Processing*, 2019, 16(5):1559-1571

[11] Wang L, Cao Y, Du G, et al. A low-latency depth modelling mode-1 encoder in 3D-high efficiency video coding standard[J]. *Journal of Electronics and Information Technology*, 2019, 41(7):1625-1632

[12] Sanchez G, Saldanha M, Fernandes R, et al. 3D-HEVC bipartition modes encoder and decoder design targeting high-resolution videos[J]. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020, 67(2): 415-427

[13] Joint Collaborative Team on 3D Video Coding Extension Development. JCT3V-K1003 Test Model 11 of 3D-HEVC and MV-HEVC[S]. Geneva: International Telecommunication Union, 2015

[14] Joint Collaborative Team on 3D Video Coding Extension Development. JCT3V-G1100 Common Test Conditions of 3DV Core Experiments[S]. San Jose: International Telecommunication Union, 2014

Xie Xiaoyan, born in 1972. She is a professor in Xi'an University of Posts and Telecommunications. She received her B.S. degree from Nanjing University of Science and Technology in 1995. She received her M.S. degree in Computer Science Department of Xidian University in 2002. Her research interests include the design of parallel computing architecture and multimedia algorithms for parallel processing.