International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 21 (2017) pp. 10895-10901
© Research India Publications. http://www.ripublication.com

Lossless Frame Memory Compression with Low Complexity using PCT and AGR for Efficient High Resolution Video Processing

Jongho Kim

Department of Multimedia Engineering, Sunchon National University,
255 Jungangro, Suncheon, Jeonnam 57922, South Korea.

Orcid: 0000-0003-4452-7587

Abstract

This paper propose a lossless frame memory compression algorithm with low complexity based on the photo core transform (PCT) with adaptive prediction and adaptive Golomb-Rice (AGR) coding for high resolution video applications. The main goal of the proposed algorithm is to reduce the amount of frame memory in video codec losslessly for high resolution video applications. The proposed method has a block-based structure in which the basic unit of processing is a macroblock (MB) of each video frame. The combination of the PCT with adaptive prediction and AGR is utilized to achieve lossless and low complexity compression. The proposed algorithm does not need any memory operation to store data, so that it is able to minimize the implementation cost. Simulation results illustrate that the proposed algorithm achieves a superior compression performance to the existing methods it is easily applicable to hardware devices without image quality degradation and with negligible structure modification. Moreover, the proposed algorithm can be employed to real-time applications in terms of its considerable improvement of the compression ratio but a slight increment of its execution time.

Keywords: frame memory compression, lossless compression, PCT, adaptive prediction, high resolution video

INTRODUCTION

Video coding techniques have been developed to meet the requirement for efficient data transmission, storage, and utilization of hardware resources. As the demand for high definition (HD) and ultrahigh definition (UHD) video applications increases and display technology advances, video codec has to process large amount of data within a bounded time [1]. In this context, High Efficiency Video Coding (HEVC) [2] was developed as an advanced successor of H.264/AVC [3]. These codecs require much larger amount of internal frame memory to deal with high resolution frames. However, the large amount of frame memory requirement usually degrades system performance and results in high energy consumption. Thus, numerous studies are aimed to alleviate this performance bottleneck. Although many previous works propose memory-efficient architectures to reduce redundant memory access for functional modules such as motion estimation (ME) and motion compensation (MC), these works are inflexible to support various video coding systems [4, 5]. Alternatively, many frame memory compression algorithms are proposed to reduce the memory traffic [6-15]. The main concept of frame memory compression is to compress video frames before being stored into the internal memory which is used for ME, MC, and so on. The frame memory compression algorithms should be easily integrated into various video coding systems, so that they are independent of memory types and video coding standards.

Frame memory compression algorithms can be either lossy or lossless. Lossy algorithms can guarantee the compression performance at the expense of video quality loss [6]. On the other hand, lossless algorithms preserve video quality [7, 8]. We can further categorize the memory compression into line-based algorithms and block-based algorithms according to how the spatial locality information of pixels is used to perform compression. A block-based algorithm takes M×N block in a frame as a basic unit and compresses pixels in a block by using the neighboring pixel information in the same block. It is mainly used for compressing reference frames, since they can provide random access ability for each M×N block. Song and Shimamoto propose an algorithm that uses a differential of adjacent pixels (DAP) and the Huffman coding scheme to compress every 8×8 block in a frame [9]. Lee et al. propose a reference frame compression scheme that contains a hierarchical minimum and difference (HMD) method to calculate the difference between pixels in an 8×8 block and encodes the differences using the Exp-Golomb coding [10]. Kim et al. propose a compression algorithm based on the analysis results for 15 1080p HD video sequences [11]. They use a hierarchical prediction and grouping method to predict 8×8 blocks and compress them using a truncated bit packing method. A line-based algorithm, on the other hand, takes one or more pixels of a video line as a basic unit and compresses pixels using only neighboring pixels in the same line. It is mainly used for compressing frame buffers in the display devices which show video frames line by line. Lei et al. propose a dictionary-based compression algorithm for display frame buffers [12]. The algorithm uses a buffer to store several distinct previous pixels as dictionary pixels and a pointer to indicate the buffer address that is to be updated next. Yang et al. propose another dictionary-based algorithm by analyzing the characteristics of display frames [13]. Yng et al. propose a line-based compression algorithm to reduce the memory size for display devices [14]. The algorithm uses a modified Hadamard transform to decorrelate the pixels in a line and compresses the coefficients using an adaptive Golomb-Rice coding method. Dikbas and Zhai propose a lossless frame compression algorithm with a fractional line-buffer [15]. It uses a (5, 3) integer wavelet transform to decorrelate pixels and compresses the coefficients using an adaptive Golomb-Rice coding method.

For high quality of digital image applications, the extended color range has becoming more important in various emerging products. Recently, digital cameras and the display devices can deal with 12 bits per channel as well as 8 bits per pixel. This leads the development of a new still image coding standard, JPEG XR (extended range), which is designed for the high dynamic range (HDR) and the HD photo size [16]. The goal of JPEG XR is to support the greatest possible level of dynamic range and color precision, and to keep the device implementations of the encoder and decoder as simple as possible. JPEG, which uses a discrete cosine transform (DCT) to decorrelate the pixels, is a widely-used image compression format because of its simplicity and coding efficiency [17]. Another image coding standard, JPEG2000, uses a discrete wavelet transform (DWT) for better coding efficiency [18]. However, the design of JPEG2000 is much complicated than the JPEG standard. JPEG XR has many coding tools including the photo core transform (PCT) and its prediction to improve the compression performance with low complexity. The PCT is a reversible transform based on integer manipulations, so that it makes the lossless and low complexity compression possible. Consequently, this paper employs the PCT as one of the core modules of the internal frame memory compression for high resolution video coding systems.

We propose, in this paper, a lossless frame memory compression algorithm with low computational complexity for high resolution video codecs. The proposed algorithm includes the block-based compression scheme to deal with high resolution video effectively and the low complexity consideration to support its real-time application. We also compress video data losslessly, since video quality of the internal frame memory of video codec can affect the performance of the whole video processing system. The proposed algorithm divides an input image into blocks with the predefined size, and applies the PCT of JPEG XR to each block for decorrelation of the pixels. Then, the predicted coefficients of PCT are provided to the adaptive Golomb-Rice (AGR) coding for the entropy coding. We describe from the comprehensive experiments that the proposed frame memory compression algorithm achieves better compression performance and lower computational complexity compared with prior arts.

PROPOSED FRAME MEMORY COMPRESSION ALGORITHM

Structure of the proposed algorithm

The overall flow of the proposed lossless and low complexity frame memory compression algorithm for efficient high resolution video processing is illustrated in Figure 1. The input of the proposed algorithm is a macroblock (MB) of 16 x 16 pixels, considering the existing video codecs, and the output is the compressed bitstream segment for the corresponding MB. The PCT applied to each MB produces one DC, 15 AD, and 240 AC components. After that, the coefficients are predicted by using the components from neighboring MBs for more reduction of data. The scanning process converts 2D distribution of the coefficients into 1D ordering, then the DC component is coded as the fixed length code (FLC), and the AD and AC components remapped by the Rice mapping are coded as the variable length code (VLC) using the adaptive Golomb-Rice (AGR) coding. Finally, the formatting process makes up the compressed bitstream segment according to specific rules.

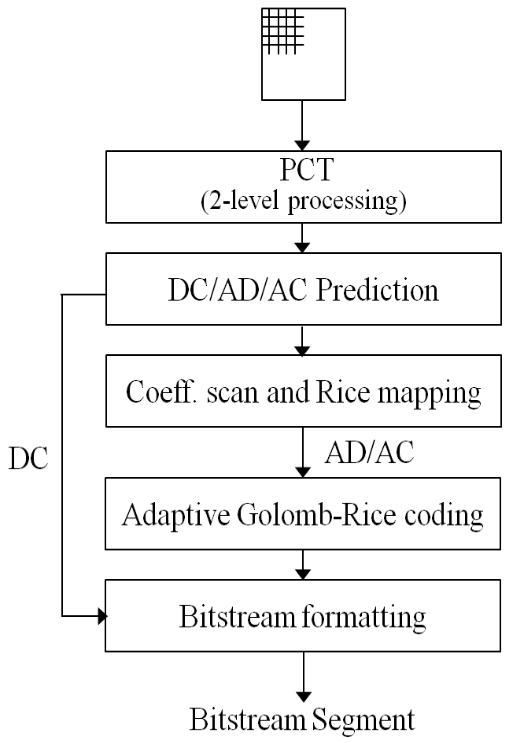
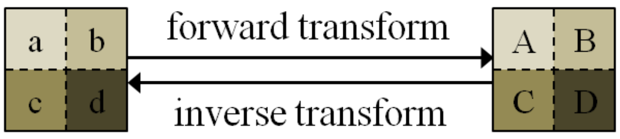


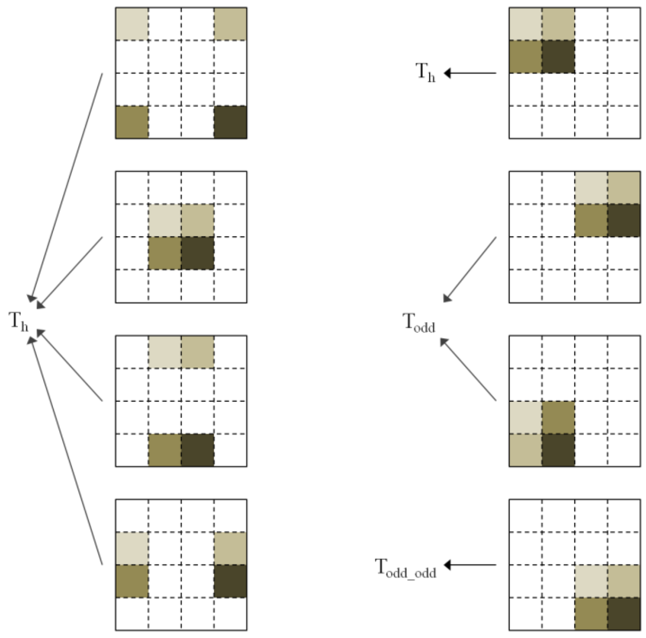
Figure 1: Overall flow of the proposed algorithm

Photo Core Transform (PCT)

The photo core transform (PCT) decorrelates the pixel values by transforming them into lowest frequency coefficient, DC, low frequency coefficients, AD, and high frequency coefficients, AC [19]. We can see in Figure 2(a) that the relationship of the forward transform and the inverse transform between pixel values and frequency components. The 2D 4×4 PCT is built by using three operators: T_h, 2×2 Hadamard transform for lowpass band, T_{odd}, 2×2 filter for low-high and high-low band, and T_{odd_odd}, 2×2 filter for high-high band, as shown in Figure 2(b).



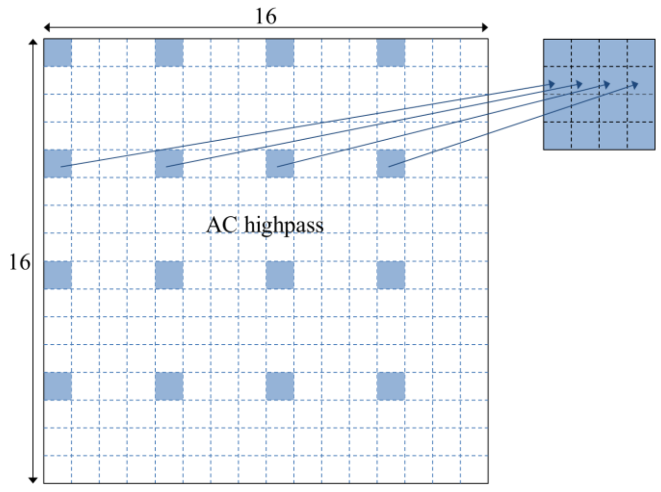
(b)



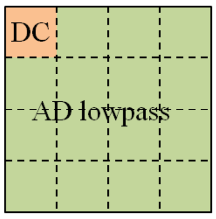
(b)

Figure 2: PCT operation, (a) relationship of the forward/inverse transform, (b) three operators for PCT: T_h, T_{odd}, and T_{odd_odd}

There are two levels for PCT as shown in Figure 3. A macroblock (MB) is partitioned into 16 blocks with 4×4 pixels, then each block is transformed by the 4×4 PCT in each level. In the first level shown in Figure 3(a), a 2×2 T_h operation is applied to four pixels by four times for each block as the left side of Figure 2(b). Three types of 2×2 transforms are then applied to the former coefficients for each block as the right side of Figure 2(b). The coefficients except a DC from the first level of PCT make up AC highpass components for a 4×4 block. After the first level, the DC coefficients of 16 4×4 blocks can be collected as a 4×4 DC block. The second level shown in Figure 3(b) is processed by applying the operations same as the first level to the 4×4 DC block, so that one DC and 15 AD lowpass components are produced for a MB.



(a)

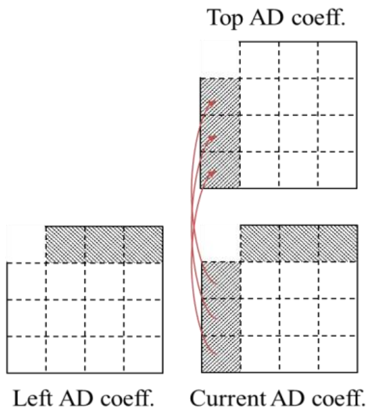


(b)

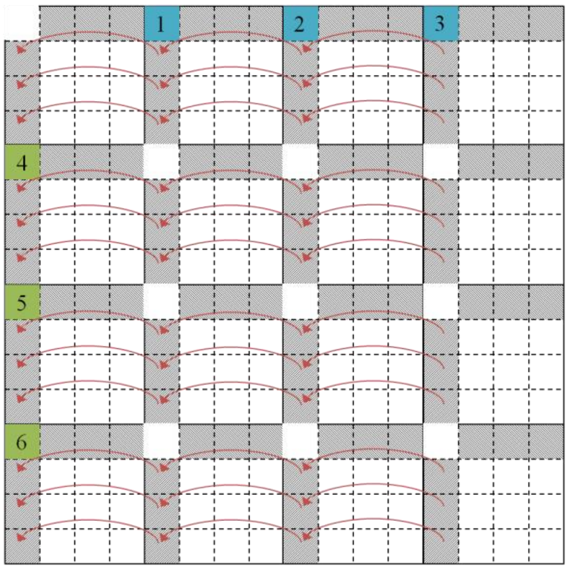
Figure 3: Two PCT levels for each MB, (a) the first level producing a DC block and AC highpass components, (b) the second level producing a DC component and AD lowpass components

Prediction of PCT coefficients

The transformed coefficients are predicted to improve the compression performance. There are three directions for DC prediction: LEFT, TOP, and LEFT_TOP [19]. The DC prediction rules use the DC coefficient of left MB and top MB for the current MB to decide the direction of the prediction. We calculate two weights: H, difference between the DC coefficients of top-left MB and top MB, and V, difference between the DC coefficients of top-left MB and left MB. DC of the current MB is predicted from DC of left MB when V is larger than four times of H; from DC of top MB when H is larger than four times of V; from left MB and top MB otherwise. The AD/AC coefficients are also predicted from coefficients of its top MB or left MB. The prediction relationship example of AD coefficients is shown as Figure 4(a), when the predicted direction is TOP. The AD predicted direction follows the DC prediction model. The AC predicted direction can be decided, after comparing the values of H and V. In this case, H can be obtained by sum of absolute values of lowpass value 1, 2, and 3 shown in Figure 4(b), and V can be obtained by sum of absolute values of lowpass value 4, 5, and 6 shown in Figure 4(b). AC predicted direction is set to LEFT when V is larger than four times of H; TOP when H is larger than four times of V; NULL prediction otherwise. The computation after prediction judgment of AC is similar to AD prediction that can be reused to reduce the coefficient value of the block. The prediction relationship example of AC coefficients is shown in Figure 4(b) when the prediction direction is LEFT.



(a)



(b)

Figure 4: Prediction of AD and AC coefficients, (a) AD coefficients predicted from TOP, (b) AC coefficients predicted from LEFT

Scanning and Rice mapping

The 2D distribution of coefficients obtained from PCT with AD and AC prediction should be rearranged into 1D ordering to provide them with AGR coding. This work accomplishes the conversion by using the scanning order introduced in JPEG XR as shown in Figure 5. There are two scanning order, one for AD and AC coefficients predicted from LEFT as shown in Figure 5(a) and the other for AC coefficients predicted from TOP as shown in Figure 5(b).

x	4	1	5
8	2	9	6
12	3	10	13
7	14	11	15

(a)

x	1	2	5
4	3	6	9
8	7	12	15
13	10	11	14

(b)

Figure 5: Scanning order, (a) for AD and AC coefficients predicted from LEFT, (b) for AC coefficients predicted from TOP

A Golomb-Rice coding accepts only a non-negative number as its input. However, the transform coefficients can be a negative number. Thus, it is required to convert the transform coefficients into a non-negative number. This is usually accomplished by Rice mapping as follows:

$$p = \begin{cases} 2n, & n \geq 0 \\ -2n-1, & n < 0 \end{cases} \qquad (1)$$

where n represents the transform coefficient and p denotes the input value of the Golomb-Rice coding. The inverse mapping from the value p back to n is very simple, since the least significant bit (LSB) of p indicates the sign of n and the magnitude of n is simply obtained by the 1-bit right shift operation with respect to p.

Adaptive Golomb-Rice (AGR) coding

We apply adaptive Golomb-Rice (AGR) coding to the coefficients that are 1D-ordered by scanning process and Rice mapping for an entropy coding [14]. Golomb-Rice (GR) coding has been used widely in modern compression systems such as JPEG-LS and H.264/AVC. The GR coding is optimal or near-optimal for integer sources with two-sided geometric distributions [20]. In other words, GR coding approximates the optimal Huffman coding for such sources, with minimal loss in efficiency. The GR coding scheme is convenient for use on a computer, since multiplication and division by 2 can be implemented using a bit-wise shift operation, which can be performed extremely quickly. Similarly, calculating the corresponding remainder can be achieved by a simple bit-wise mask operation, which is a very fast operation. Thus, the GR coding is used for an entropy coding in a number of lossless and lossy image compression methods. The GR coding requires less computational power than Huffman coding, and it can be easily utilized in an adaptive fashion whereas adaptive Huffman coding requires very large amount of computational power.

The flow of the adaptive Golomb-Rice (AGR) coding for each MB is shown in Figure 6. Each MB has different *k* value for the maximum compression performance. In order to determine the optimal parameter *k*, the lengths of GR coded bits for all AD and AC components are summed up. The length of the GR code for a given value *p* is calculated by

$$l_{GR} = k + 1 + p / 2^k \tag{2}$$

The best *k* value is determined based on the minimum sum of the GR code lengths. For a small value, a smaller *k* leads to a smaller GR code length. As the value increases, a larger *k* may produce a smaller code length. Thus, the choice of *k* depends on the value adaptively.

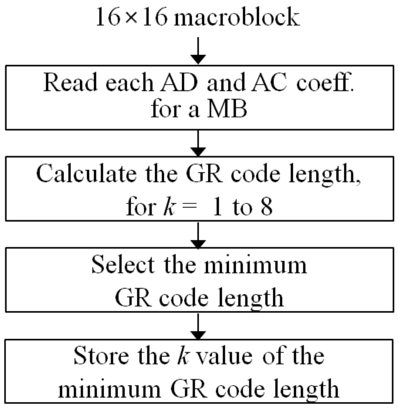

16×16 macroblock
Read each AD and AC coeff. for a MB
Calculate the GR code length, for k = 1 to 8
Select the minimum GR code length
Store the k value of the minimum GR code length

Figure 6: Flow diagram for AGR coding

Bitstream Formatting

The AGR codes for AD and AC components are arranged as a bitstream segment as shown in Figure 7. The *k* value for each MB is coded by the 3-bit fixed length code (FLC) and stored in the leftmost position. Next, the DC and AD components from the second level PCT are stored with the 8-bit FLC and the AGR codes, respectively. Once all the DC and AD components of the second level PCT are stored in the bitstream, the remaining AC components from the first level PCT are coded and appended. The resulting bitstream format is illustrated in Figure 7.

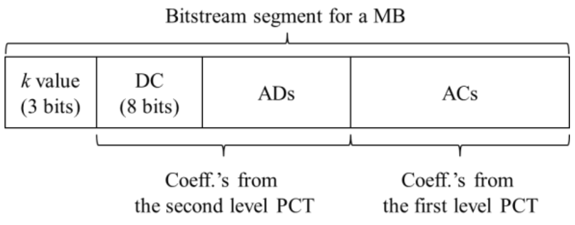

Bitstream segment for a MB
k value (3 bits)
DC (8 bits)
ADs
ACs
Coeff.'s from the second level PCT
Coeff.'s from the first level PCT

Figure 7: Bitstream segment format for each MB

EXPERIMENTAL RESULTS

We evaluate the performance of the proposed algorithm and compare our algorithm with the existing low complexity and lossless compression methods. The existing methods are Song's method, the differential of adjacent pixel (DAP) with Huffman coding proposed in [9], Lee's method, the modified Hadamard transform (MHT) with Golomb-Rice coding as proposed in [6], and Yng's method, the MHT with AGR coding as proposed in [14]. However, Lee's method is a lossy compression, and Yng's method has a line-based structure. For fair comparisons, we remove the quantization process to make Song's method perform lossless compression, and modify Yng's method into a block-based compression structure in our simulations.

We simulate the proposed algorithm including PCT and AGR coding for each basic unit of MB as summarized in Figure 8. Then, we compare the results with the existing algorithms. Also, we analyze computational complexity by measuring the execution times of the existing algorithms and the proposed algorithm. The comparison results of the coding performance of each algorithm are shown in Table I. In order to obtain an enough empirical evaluation of the proposed low complexity and lossless compression algorithm, we use a set of well-known test images. In our experiments, we use 8 widely-used images with 512×512 pixels, which are listed in the first column of Table I. The average compression ratio for each algorithm is also described in the table. As a measure of the compression performance, we use the measure of the compression ratio (CR) as in (3).

$$\mathrm{CR} = \left(1 - \frac{\text{Compressed Size}}{\text{Original Size}}\right) \times 100\ (\%) \tag{3}$$

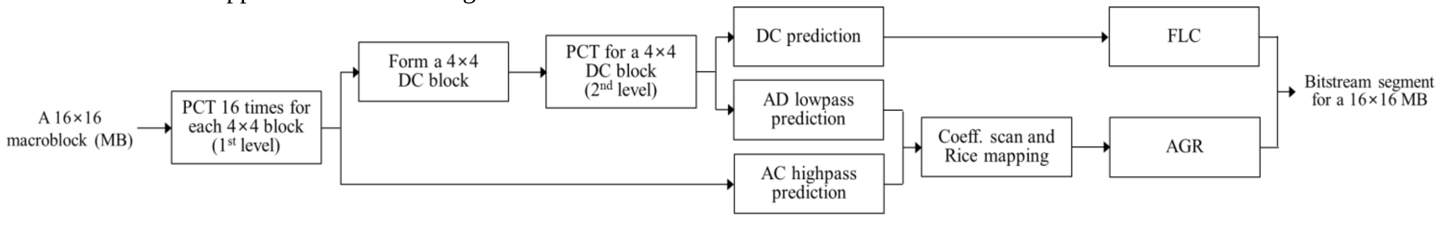

A 16×16 macroblock (MB)
PCT 16 times for each 4×4 block (1st level)
Form a 4×4 DC block
PCT for a 4×4 DC block (2nd level)
DC prediction
AD lowpass prediction
AC highpass prediction
Coeff. scan and Rice mapping
FLC
AGR
Bitstream segment for a 16×16 MB

Figure 8: Overall processing summary of the proposed algorithm

Table 1 indicates that the proposed method are superior to the existing methods in terms of coding efficiency. We can observe an average of 15% increment in terms of the CR for the proposed algorithm as compared to the reference algorithms. Specifically, the proposed algorithm improves the compression ratio by 16% on average compared with Song's method [9], by 18% on average compared with Lee's method [6], and by 9.8% on average compared with Yng's method [14] which is made up of MHT and AGR coding.

Table 1: CR values of the reference and the proposed algorithms for comparison of coding efficiency (in %)

	Song	Lee	Yng	Proposed
Airplane	35.9	33.3	38.8	41.3
Baboon	17.7	17.2	20.1	24.6
Barbara	26.8	28.9	28.2	30.7
Elaine	28.8	29.1	33.1	37.5
Lena	30.3	31.1	35.7	39.4
Man	26.4	25.3	27.2	31.1
Peppers	32.6	27.2	33.4	37.3
Zelda	39.0	40.3	41.4	44.0
Average	29.7	29.1	32.2	35.7

The analysis results of computational complexity for the conventional methods and the proposed algorithm are shown in Table 2. The computational complexity is measured by execution time of software implementation for each method. The simulation is carried out on QuadCore CPU with 2.2 GHz machine and the software is implemented with C++ language. We execute the software 10 times for each image and average the execution times.

Table 2: The execution times of the reference and the proposed algorithms for comparison of computational complexity (in ms)

	Song	Lee	Yng	Proposed
Airplane	27.3	22.7	25.5	26.7
Baboon	31.0	25.3	29.7	32.8
Barbara	32.1	26.6	30.8	31.1
Elaine	26.7	20.1	23.3	24.3
Lena	27.6	21.2	25.1	25.5
Man	27.3	24.8	27.9	29.2
Peppers	25.4	23.9	25.9	28.4
Zelda	24.9	21.4	23.4	24.5
Average	27.8	23.3	26.5	27.8

It can be found from Table 2 that the execution times of the proposed algorithm are similar to those of the conventional methods or are increased slightly. Although the execution time of Lee's method [6] is shortest among the conventional and the proposed methods, it has a problem with insufficient compression performance. Song's method [9] is limited to application due to computational cost and additional memory requirement for Huffman coding. Yng's method [14] is competitive with the proposed algorithm in terms of the compression performance and execution time. However, the proposed method outperforms Yng's method by 9.8% on average of CR, even though the execution time of our method is increased by 4.7% on average, which does not influence much on the operation complexity.

CONCLUSIONS

This paper has presented a lossless frame memory compression algorithm with low complexity. The main purpose of the proposed algorithm is to reduce the amount of frame memory in video codec losslessly for high resolution video applications. In order to achieve the aim, we have proposed the combination of the photo core transform with adaptive prediction and adaptive Golomb-Rice coding. We have selected three of the lossless and low complexity algorithms for comparison. For the evaluation of the proposed algorithm, we have obtained the average compression ratio and compared it with the conventional methods using eight well-known test images to get a fair and effective evaluation. Simulation results indicate that the proposed algorithm provides better compression ratio compared to the reference algorithms and it can be applicable to most of the hardware devices without image quality degradation and with negligible structure modification. Moreover, the proposed algorithm does not need any memory operation to store data, so that it is able to minimize the hardware implementation cost. It is also possible that the proposed algorithm is adopted to real-time applications from its considerable improvement of the compression ratio but a slight increment of its execution time compared to the reference methods. From the simulation results, future research may be directed to improve the proposed algorithm by effective prediction and transform scheme for better compression performance or efforts for the reduction of execution time.

REFERENCES

[1] Lian, X., Liu, Z., Zhou, W., and Duan, Z., 2016, "Lossless frame memory compression using pixel-grain prediction and dynamic order entropy coding," IEEE Trans. Circuit Syst. Video Technol., 26(1), pp. 223-235.

[2] Sullivan, G. J., Ohm, J., Han, W. J., and Wiegand, T., 2012, "Overview of the high efficiency video coding

(HEVC) standard," IEEE Trans. Circuit Syst. Video Technol., 22(12), pp. 1649-1668.

[3] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A., 2003, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuit Syst. Video Technol., 13(7), pp. 560-576.

[4] Tuan, J., Chang, T., and Jen, C., 2002, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," IEEE Trans. Circuit Syst. Video Technol., 12(1), pp. 61-72.

[5] Tsai, C., Chen, T., Chen, T., and Chen, L., 2005, "Bandwidth optimized motion compensation hardware design for H. 264/AVC HDTV decoder," Proc. 48th IEEE Midwest Symposium on Circuits and Systems, pp. 1199-1202.

[6] Lee, T., 2003, "A new frame-recompression algorithm and its hardware design for MPEG-2 video decoders," IEEE Trans. Circuit Syst. Video Technol., 13(6), pp. 529-534.

[7] Shen, Z., Miao, C., and Zhang, Y., 2013, "Memory bandwidth reduction for video decoders based on data arrangements," Proc. 6th IEEE International Congress on Image and Signal Processing (CISP), pp. 31-35.

[8] Cheng, C., Tseng, P., and Chen, L., 2009, "Multimode embedded compression codec engine for power-aware video coding system," IEEE Trans. Circuit Syst. Video Technol., *19*(2), pp. 141-150.

[9] Song, T., and Shimamoto, T., 2007, "Reference frame data compression method for H. 264/AVC," IEICE Electronics Express, 4(3), pp. 121-126.

[10] Lee, S., Chung, M., Park, S., and Kyung, C., 2009, "Lossless frame memory recompression for video codec preserving random accessibility of coding unit," *IEEE Trans. Consum. Electron.,* 55(4), pp. 2105-2113.

[11] Kim, J., Kim, J., and Kyung, C., 2009, "A lossless embedded compression algorithm for high definition video coding," Proc. 2009 IEEE International Conference on Multimedia and Expo (ICME), pp. 193-196.

[12] Lei, J., Zou, X., Wu, Z., and Fan, W., 2007, "Research of an image map encoding algorithm on frame buffer," Proc. 7th IEEE International Conference on ASIC (ASICON), pp. 894-897.

[13] Yang, H., Chen, J., Kuo, H., and Lin, Y., 2009, "An effective dictionary-based display frame compressor," Proc. IEEE/ACM/IFIP 7th Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia2009), pp. 28-34.

[14] Yng, T., Lee, B., and Yoo, H., 2008, "A low complexity and lossless frame memory compression for display devices," *IEEE Trans. Consum. Electron.,* 54(3), pp. 1453-1458.

[15] Dikbas, S., and Zhai, F., 2010, "Lossless image compression using adjustable fractional line-buffer," Signal Process.: Image Commun., 25(5), pp. 345-351.

[16] Dufaux, F., Sullivan, G. J., and Ebrahimi, T., 2009, "The JPEG XR image coding standard," IEEE Signal Process. Mag., 26(6), pp. 195-199.

[17] Pennebaker, W. B., and Mitchell, J. L., 1992, JPEG: Still Image Data Compression Standard, Van Nostrand Reinhold, NY.

[18] Diego, S., and Ebrahimi, T., 2000, "An analytical study of JPEG 2000 functionalities," Proc. IEEE International Conference on Image Processing, 2, pp. 49-52.

[19] Pan, C., Chien, C., Chao, W., Huang, S., and Chen, L., 2008, "Architecture design of full HD JPEG XR encoder for digital photography applications," IEEE Trans. Consum. Electron., 54(3), pp. 963-971.

[20] Merhav, N., Seroussi, G., and Weinberger, M. J., 2000, "Optimal prefix codes for sources with two-sided geometric distributions," IEEE Trans. Inform. Theory, 46(1), pp. 121-135.