Intra-picture Block-matching Method for Codebook-based Texture Compression

Li Cui[1] and Euee S. Jang[2]
[1] Department of Computer & Software, Hanyang University
Seoul, Republic of Korea;
and Department of Computer Science & Technology, Yanbian University
Yanji, China;
[e-mail: Lcui2000@gmail.com]
[2] Department of Computer & Software, Hanyang University
Seoul, Republic of Korea
[e-mail: esjang@hanyang.ac.kr]
*Corresponding author: Euee S. Jang

Received March 30, 2016; revised August 5, 2016; accepted September 18, 2016; published October 31, 2016

Abstract

In this paper, an efficient texture compression method is proposed for fast rendering, which exploits the spatial correlation among blocks through intra-picture block matching. Texture mapping is widely used to enhance the visual quality of results in real-time rendering applications. For fast texture mapping, it is necessary to identify an effective trade-off between compression efficiency and computational complexity. The conventional compression methods utilized for image processing (e.g., JPEG) provide high compression efficiency while resulting in high complexity. Thus, low complexity methods, such as ETC1, are often used in real-time rendering applications. Although these methods can achieve low complexity, the compression efficiency is still lower than that of JPEG. To solve this problem, we propose a texture compression method by reducing the spatial redundancy between blocks in order to achieve the better compression performance than ETC1 while maintaining complexity that is lower than that of JPEG. Experimental results show that the proposed method achieves better compression efficiency than ETC1, and the decoding time is significantly reduced compared to JPEG while similar to ETC1.

Keywords: texture compression, spatial correlation, vector quantization, codebook

This research was supported by the International Research & Development Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2015K1A3A1A21000259).

http://dx.doi.org/10.3837/tiis.2016.10.023 **ISSN : 1976-7277**

1. Introduction

Texture compression is widely used in real-time rendering applications to reduce the size of texture images while decreasing rendering performance. Many compression methods have been proposed for images, most of which are mainly employed for compression for either storage or transmission. However, it is necessary for a texture compression system to allow fast random access to texture data. Conventional compression methods, such as JPEG, are not usually utilized for texture compression, because it is computationally expensive to perform random access decoding. Moreover, JPEG produces variable length codes for entropy coding, which results in a large portion of the texture image being required for decoding a texture pixel.

Several texture compression methods to address random access during rendering time have been proposed, which can be classified into: 1) block-based methods; and 2) vector quantization (VQ)-based methods. Block-based compression methods have constituted a frequent area of research [1]-[7]. S3 Texture Compression (S3TC) (called DXTC in DirectX) is a local palette technique, which uses 4×4 or 8×8 pixel blocks and achieves four bits per pixel (*bpp*) [2]. Jacob et al. proposed Ericsson Texture Compression (ETC), and claimed that it can simplify hardware implementation by compressing 2×4 pixel blocks into 32 bits per block [3]. An extension of ETC (called ETC1) is proposed, which splits a texture into 4×4 pixel blocks [4]. And ETC2 adds three nodes to handle chrominance edges better than ETC1 [5]. ETC1 and ETC2 can provide better quality than S3TC and PVRTC at the cost of a decrease of compression ratio, which are fixed-rates methods. Moreover, several block-based methods have been proposed to provide alternative formats that allow variable bit-rate texture compression [6][7]. Although low complexity is achieved by these schemes, their compression efficiency is less than that of JPEG. A challenging task of block-based methods is identifying a proper trade-off between compression efficiency and computational complexity for fast rendering. VQ-based methods have been proposed by several researchers [8][9]. For example, Beers et al. first proposed that VQ can be applied to texture compression because textures usually contain many repeating patterns [9]. Texture color tables based on VQ are also efficiently employed for hardware-assisted decoding of vector components in [10]. And Yi Xiao et al. used self-orgnizing map algorithm to construct large color palette based on VQ for HDR texture compression [11]. We observe that most of the VQ-based methods are based on the color palette as a form of VQ, but exploiting the similarity between blocks has rarely been considered to form a VQ codebook for texture compression.

The block-matching algorithm (BMA) has long been utilized in numerous fields. It is used to find the closest block among a set of blocks by comparing the given block with blocks within a given search range. BMA is commonly employed to estimate motion vectors for the improvement of inter-frame coding performance in video coding. Moreover, BMA can also be applied for intra-frame coding. Intra-block copy (IntraBC) as a BMA for intra-frames has been recently adopted into the extension of High Efficiency Video Coding (HEVC) for screen-content coding [12]-[14]. The HEVC encoder requires low computational complexity for real-time applications, which differentiates it from the texture encoder. Thus, IntraBC exploits spatial redundancy within a specified search range.

In the present paper, we propose an efficient texture compression method to exploit the spatial correlation between blocks by using the intra-picture block-matching method. In the proposed method, the blocks in the original texture image are characterized by a small set of

selected blocks (called a codebook) based on the distortion measured by the block-matching method. In order to take advantage of the spatial correlation of blocks in the proposed method, high compression efficiency is expected for the texture images maintaining high similarities among blocks.

The rest of the paper is organized as follows. In section 2, we formulate the posed problem in a mathematical manner. Section 3 presents the proposed codebook-based texture compression method in detail. Experimental results and comparisons with other conventional methods are presented in section 4. Finally, conclusions are drawn in section 5.

2. Problem definition

Let us assume that there are N blocks per texture image, and M blocks among them are chosen to represent N blocks. If we use the fewer blocks (i.e., $M<N$), the total bit cost based on M (C_{Total}) is smaller than that of the original design by using N blocks. Thus, the relationship between M and C_{Total} is as follows:

$$C_{Total} = M \times V \tag{1}$$

where V indicates the average number of bits used in a block. Equation (1) implies that minimizing M can achieve low C_{Total} at the cost of high distortion of replaced blocks. Thus, the total distortion between the replaced blocks and the original blocks (D_{Total}) has to be considered to estimate the total bit cost. This relationship can be described using the Lagrange equation of cost (E) based on M and D as follows:

$$E = M \times V + kD_{Total} \tag{2}$$

where k is a Lagrange multiplier. The optimal selection of M is a combinatorial problem which is NP-complete. As the number of N increases, it is very difficult to obtain an optimal solution in a reasonable computation time. For practical applications, we need an efficient heuristic method to find an approximate solution. To this end, an acceptable threshold (Th) is defined as a limitation for the distortion of each block (P). While $P \leq Th$, the optimal selection of M is given by minimizing D_{Total}, defined as:

$$M_{optimal} = \underset{i\in[1,N]}{\operatorname{argmin}} E(i) \approx \underset{i\in[1,N]}{\operatorname{argmin}} D_{Total}(i)$$

$$D_{Total}(M) = \sum_{i=1}^{M} S_i P_i$$

subject to the following constraints:

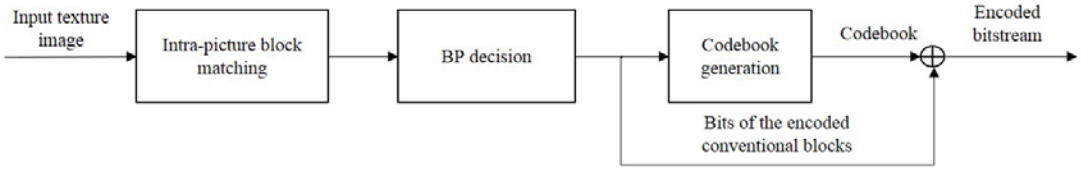
$$\sum_{i=1}^{M} S_i = N \text{and } P_i \leq Th \tag{3}$$

where S_i is the number of blocks replaced by the i-th selected block; and P_i indicates the distortion between the ith selected block and the replaced block at the corresponding position. The minimum of D_{Total} can be achieved through comparing the distortions among the different combinations of M blocks. Thus, the minimum of the total cost (E_{min}) based on $M_{optimal}$ is given by:

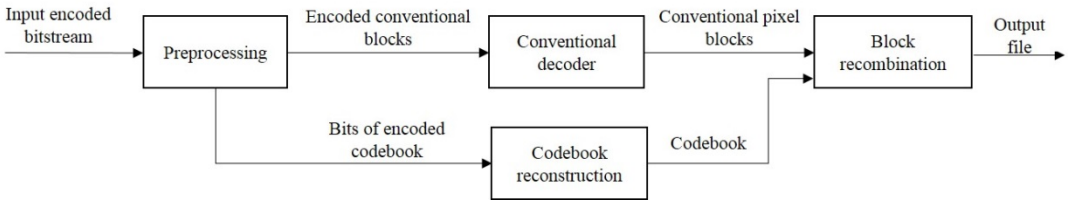
$$E_{min} = M_{optimal} \times V + k \sum_{i=1}^{M_{optimal}} S_i P_i \tag{4}$$

3. Proposed method

To achieve high compression efficiency while maintaining acceptable complexity, we propose a texture compression method to reduce the spatial redundancy between blocks by using the block-matching method. The proposed block-matching method is utilized to compute the distortions (P) on a block-by-block basis and to measure the spatial correlation among blocks by comparing their distortions with an acceptable threshold (Th). In order to obtain the local minimum of D_{Total} as given in Equation (4), a heuristic process, termed the block-pattern (BP) decision process, is performed. A high-level block diagram of the proposed method is given in **Fig. 1**.



(a) Encoder



(b) Decoder

Fig. 1. Block diagram of the proposed method

3.1. Encoding process

As shown in **Fig. 1** (a), the encoding process of the proposed approach can be summarized in the following three steps. In the first step, the pixel-level similarity between each block and the rest of the encoded blocks is measured through the intra-picture block-matching process. All of the blocks are then characterized by a small set of the conventional blocks using VQ based on the results of the previous step. Thus, the BP of each block is determined. BP indicates that the corresponding block is encoded by the conventional method or replaced by its index. Finally, a codebook for the blocks of the original texture image is generated by following the results of the block-matching and BP decision processes.

A. Block-matching process

To exploit the spatial correlation among blocks, the intra-picture block-matching process for each block is proposed to measure the distortion between blocks. The flow chart of the intra-picture block-matching process is presented in **Fig. 2**.

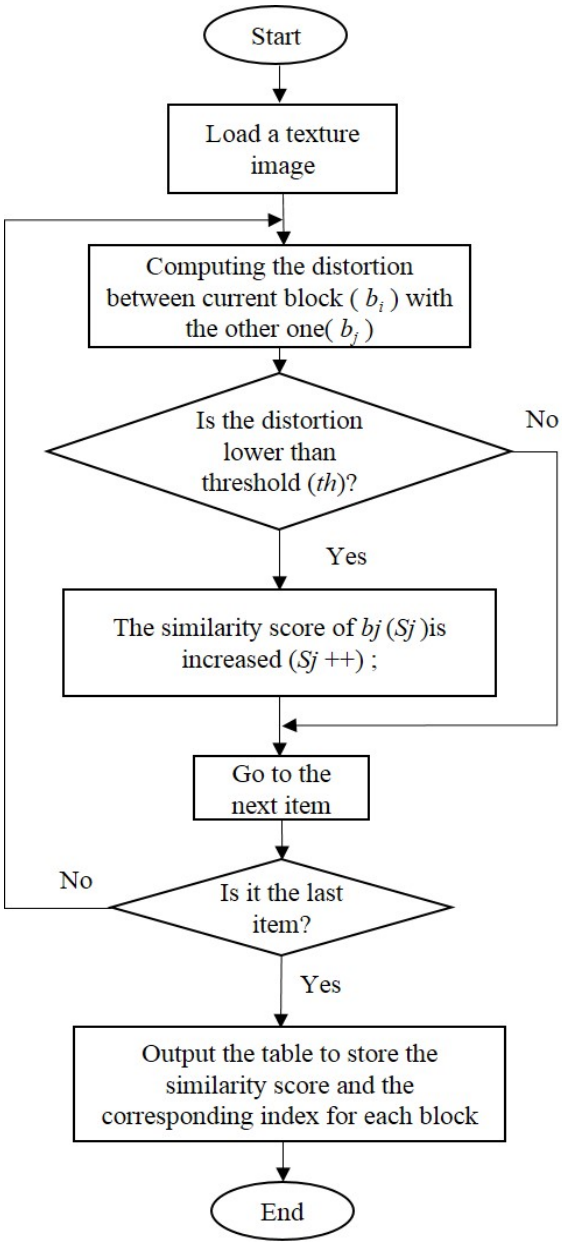


Fig. 2. Flow chart of the block-matching process

After an input texture image that consists of N blocks ($b_0 \ldots b_{N-1}$) is loaded, the block-matching process begins by measuring the distortion (P_j) between the current block and the other reference block, and it is compared with a specified *Th*. The similarity score of the current block (S_j) indicates the number of blocks whose distortions are lower than *Th*. As the result of the block-matching process, those blocks having S_j larger than 1 can be used to replace corresponding blocks. Thus, a small set of blocks (the number of blocks being defined as M) is generalized by removing those replaced blocks. M blocks are used to approximately represent the original texture image, and the total distortion (D_{Total}) compared with the original image can be described as:

$$D_{Total|Th} = S_0P_0 + S_1P_1 + S_2P_2 + \cdots + S_{M-1}P_{M-1}, \quad \mathrm{M} \leq \mathrm{N} \tag{5}$$

As the proposed block-matching process determines *M* blocks from the original *N* blocks according to *Th*, it is possible to find a sub-optimal solution based on the small set of blocks.

B. BP decision

As described in Equation (3), it is necessary to find a sub-optimal *M* to achieve the local minimizing D_{Total} based on the small set of blocks determined by the block-matching process. Thus, the BP decision process is performed to obtain the optimal *M* value. This process determines each block as the one to be encoded by using the conventional compression method, or as the one to be replaced by the index of the corresponding block. The encoded block based on BP is comprised of two types: 1) the independent block (IB), that is encoded by using the conventional compression method; and 2) the repetition block (RB), that is to be replaced by one of the IBs.

The BP decision process is performed based on the results of the block-matching process. First, the block maintaining the highest similarity score has the highest spatial correlation with other blocks. Thus, the block is determined as IB, and its index is used to replace those RBs associated with it. Next, those blocks whose patterns have been determined are excluded from the following decision process. Finally, the previous process is iteratively performed until the BPs of all blocks are determined.

C. Codebook structure

After the BPs of all blocks are determined for an input texture image, a codebook is generated to represent the necessary attributes for each block. As shown in **Fig. 3**, the codebook is formed by *N* codewords. The codeword for IB is represented by a Boolean value for BP (*bp*). The codeword for RB includes not only its *bp* value, but also the index value of the IB being used to replace it. The range of the index value for the *i*th block is between zero and the sum of IBs up to the (*i*-1)th block. The index value is encoded using a variable length code, whose bit length is computed based on the sum of IBs.

Index of codeword	**Sum of IBs**	**Codebook**		**Codeword Size**
		bp	**Index**	
0	0	0	-	1
1	1	1	0	2
2	1	0	-	1
3	2	0	-	1
4	3	1	1	3
⋮	⋮	⋮	⋮	⋮
i	*sumOfIBs(i-1)*	1	*value* ∈ [0, *sumOfIBs(i-1)*]	$\lceil \log_2 sumOfIBs(i-1) \rceil + 1$
⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮
N-1	*M-1*	*0 or 1*	none or *value* ∈ [0,(M-1)])	1 or $\lceil \log_2 (M-1) \rceil + 1$

Fig. 3. Example of codebook design based on the proposed method

It can be seen from **Fig. 3** that the total length of the codebook ($S_{codebook}$) has a linear relationship with M as follows:

$$S_{codebook} = N + \sum_{j=0}^{N-1} (p_j \times \log_2 a_j) \quad (6)$$

where p_j is the bp value of the jth block; and a_j indicates the number of IBs counted from the first block to the (j-1)th block, and the maximum of a_j is M-1. If the jth block is RB, p_j is 1; thus, the bit length of the corresponding index data is computed based on a_j. Otherwise, the jth block is IB, and the length of the corresponding index is zero for p_j = 0.

D. Encoded result

The encoded result of each texture image includes a codebook and an encoded bitstream, as shown in Fig. 4. $S_{codebook}$ has been defined in Equation (6). The size of the header part in the bitstream is small, which is compared with the length of the IB part; thus, the bits per pixel (R) is defined as follows:

$$R = \frac{M \times b + S_{codebook}}{w \times h} \quad (7)$$

where b represents the number of bits for each IB; and $w \times h$ is the resolution of the texture image.

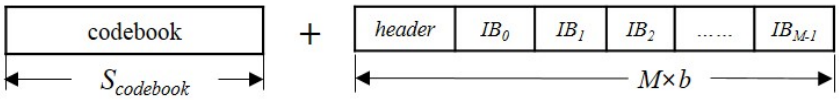


Fig. 4. Bitstream structure of the proposed method

R is calculated from Equation (7) with M=N, which is slightly increased compared with that of the conventional method ($N \times b/w/h$). Nevertheless, we can pre-compute the critical value of M based on Equations (6) and (7) in order to achieve an acceptable R. For example, b is 64 bits when ETC1 is used as the conventional compression method. And the texture image is a 512×512 24-bit RGB color image as well as N is 16384. Thus, the critical value of M is 16168, which is computed based on Equation (7). This means that R can be lower than 4 bpp (the bits per pixel by using ETC1), while M is lower than 16168. In summary, as more blocks are determined as RBs, a higher compression ratio can be achieved.

3.2. Decoding process

The decoding process of the proposed method aims to decode the corresponding block pixel based on the codebook, which can be summarized as a three-step process as shown in **Fig. 1** (b). In the first step, the encoded codebook is reconstructed and stored in memory. Then, the encoded conventional blocks are decoded by using the conventional compression method. Finally, the decoded blocks are combined into the output texture image based on the decoded codebook.

Based on the bitstream structure of proposed method shown in **Fig. 4**, we observe that computation complexity may be slightly increased by accessing the codebook. A linear relationship is defined between the decoding time of the proposed method (T_p) and the decoding time of the conventional compression method (T_c), as denoted as follows:

$$\begin{aligned} T_c &= M_t + B_t \\ T_p &= M_f + T_c \end{aligned} \quad (8)$$

where M_t represents the time of memory access for blocks; B_t is the time consumed by decoding blocks; and M_f represents the time of the memory access for BP and the index data of the blocks from the codebook. T_p is slightly increased compared to T_c because of M_f.

4. Experimental results

4.1 Test conditions

To investigate the performance of the proposed method, we used ETC1 as the compression method and implemented our proposed method based on [15]. **Fig. 5** shows the six 512×512 24-bit RGB color images used in our experiment, which include three natural images (i.e., Lena, Airplane, and Pepper), as well as three texture images (i.e., Obese, RedWood, and WoodTile). In the block-matching process, the thresholds are set to be seven different points in order to discern the tendency of performance change based on the different mean squared error (MSE) values.

Fig. 5. Six test images

4.2 Experimental results and analysis

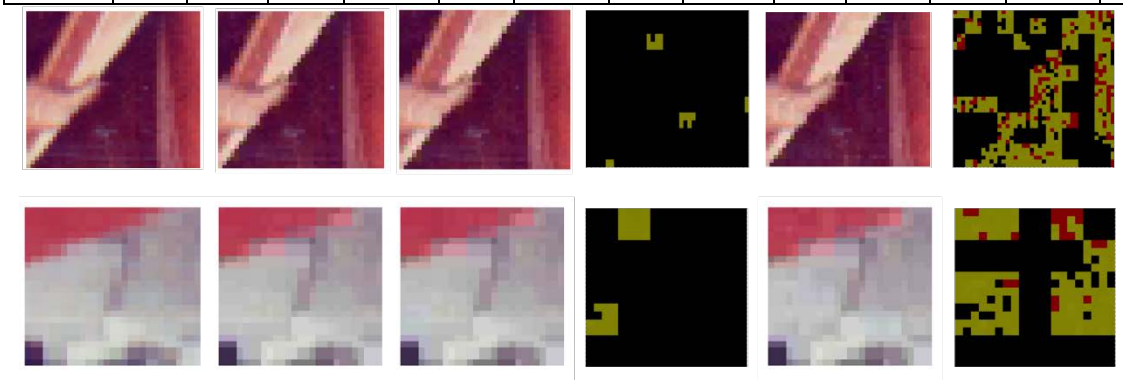
The first experiment is carried out to evaluate the rate distortion and decoding time performance of the proposed method used for the test images. As shown in **Table 1**, the RD performance of the proposed method is compared with that of the ETC1 method. The bitrate of the images containing more spatial redundancy (Airplane, RedWood, and WoodTile) is reduced faster than that of the other images while retaining good quality. Therefore, it can be concluded that the proposed method can reduce the bitrate of non-complex images faster than that of complex images. **Fig. 6** shows two examples of compressed texture images, which shows how much the proposed method affects the quality by using different thresholds. Texture images compressed by ETC1 and the proposed method based on thresholds as 200 and 400 are provided as well as the original cutouts of the compressed textures and the block maps. The original cutout is 40×40 24-bit RGB color image for Lena and 20×20 24-bit RGB color image for Airplane. The block map tells which blocks are replaced in the proposed method using different colors. The black color means that the corresponding block is not replaced. The yellow color indicates that the corresponding block is replaced as well as the red, but the red means the difference is more than the yellow.

In **Table 2**, the average decoding time of the six test images to be decoded 300 times by the proposed method is compared with that of JPEG and ETC1. To address the random access of texture data, JPEG has to decode a region of an image through decoding the whole image due to the DCT transform applied. The decoding time of JPEG is the block number times of the time to decode the whole image when the random access is conducted. Thus, it is obvious that the results of JPEG are much larger than those of ETC1 and the proposed method. The decoding time of the proposed method is almost similar to that of ETC1, which is consistent with that of Equation (8).

The second experiment is to evaluate the quality of the rendered model by using texture image. For texture compression method, Griffin et al. reported that the performance evaluation by using final rendered images matches how human perceive image distortions [11]. Thus, we rendered a 3D Model – ObeseMale by using the Obese test image. The viewpoint space for the single-object models in [16] is sampled by using the quasi-random Sobol sequence generator, but finding the expected viewer locations is left for future work. To capture the object model as clearly as possible, we fix the Y-coordinate of the viewpoints as an acceptable value. In the XZ-plane, the viewpoint space is uniformly sampled into ten viewpoints to form a set of evaluation viewpoints. The rendered results using the compressed Obese test image are captured at each viewpoint, and they are compared each to a ground truth using the uncompressed Obese test image from the same viewpoint. The average PSNR, CIE94 and RMS performance are measured as shown in **Table 3**. The results show that the texture images in the rendering stage can achieve better quality compared with that of the texture images shown in **Table 1**. This implies that the bitrate of the compression methods can be reduced while still maintaining the rendered quality.

Table 1. Comparison of PSNR and bpp between the proposed method and ETC1.

Method	*Th*	Lena		Pepper		Airplane		Obese		RedWood		WoodTile	
		PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp
ETC1		34.75	4.00	31.23	4.00	34.39	4.00	40.36	4.00	39.13	4.00	41.02	4.00
Proposed	50	34.69	3.95	30.76	4.04	34.00	3.06	39.91	3.21	39.13	4.06	40.04	3.89
	100	34.51	3.68	30.72	3.96	33.86	2.66	39.44	2.93	39.02	4.01	39.59	3.31
	200	33.97	3.07	30.55	3.63	33.62	2.28	38.34	2.50	37.05	3.18	37.25	2.16
	400	33.11	2.27	30.14	2.93	33.20	1.98	36.17	2.02	37.05	3.18	37.25	2.16
	800	31.70	1.66	28.88	2.07	32.03	1.64	33.58	1.60	37.05	3.18	37.25	2.16
	2000	28.75	1.16	27.30	1.35	29.41	1.20	30.33	1.13	37.05	3.18	37.25	2.16
	5000	25.33	0.77	24.47	0.95	26.43	0.79	25.77	0.81	37.05	3.18	37.25	2.16



Original cutout | **ETC1** | **Proposed (*Th*=200)** | **block map (*Th*=200)** | **Proposed (*Th*=800)** | **block map (*Th*=800)**

Fig. 6. Two examples of the proposed method applied to Lena (Top) and Airplane (Bottom)

Table 2. Comparison of average decoding time.

Method	Decoding Time (*ms*)
JPEG	7.17×10^4
ETC1	1.79
Proposed	2

Table 3. Average Color difference Comparison of the rendered results for ObeseMale.

Method	*Th*	PSNR	CIE94 (%)	RMS
ETC1		43.08	9.64	3.11
Proposed	50	42.47	9.86	3.33
	100	41.82	10.12	3.58
	200	40.52	10.72	4.16
	400	35.87	12.67	7.26
	800	35.69	14.18	8.89
	2000	32.11	18.29	10.96
	5000	28.60	25.08	16.41

5. Conclusion

This paper presents a codebook-based texture coding method that exploits the spatial redundancy between blocks. The experimental results demonstrate that the proposed method can achieve a flexible bitrate based on specified thresholds. Furthermore, an effective trade-off between compression ratio and computational complexity can be preserved by the proposed method, compared to that of the conventional compression method. To improve our proposed approach, further investigation can be conducted to determine an appropriate threshold value for all textures in general. And the different quality metrics may be considered to compute the difference error between blocks.

References

[1] S. Fenny, "Texture Compression Using Low-frequency Signal Modulation," in *Proc. of ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pp. 84-91, Switzerland, 2003. Article (CrossRef Link).

[2] S3TC DirectX 6.0 Standard Texture compression, White paper, S3 Corp. Article (CrossRef Link).

[3] J. Ström, and T. Akenine- Möller, "PACKMAN: Texture Compression for Mobile Phones," in *Proc. of ACM SIGGRAPH 2004 Sketches*, pp. 66, New York USA, 2004. Article (CrossRef Link).

[4] J. Ström, and T. Akenine- Möller, "iPACKMAN: High-Quality, Low-Complexity Texture Compression for Mobile Phones," in *Proc. of ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pp. 63-70, Los Angeles CA, July 2005. Article (CrossRef Link).

[5] J. Ström, and M. Pettersson, "ETC2: texture compression using invalid combinations," in *Proc. of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symp. Graph. Hardware*, pp. 49-54, 2007. Article (CrossRef Link).

[6] J.Nystad, A. Lassen, A. Pomianowski, S. Ellis and T. Olson, "Adaptive Scalable Texture Compression," in *Proc. of the 4th ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics. Eurographics Association, Aire-la-Ville, Switzerland*, pp. 105-114, 2012. Article (CrossRef Link).

[7] P. Krajcevski, A. Golas, K. Ramani, M. Shebanow and D. Manocha, "VBTC: GPU-Friendly Variable Block Size Texture Encoding," *Computer Graphics Forum*, vol. 35, pp. 409-419, 2016. Article (CrossRef Link).
[8] J. Torborg., and J.T. Kajiya, "Talisman: Commodity Realtime 3D Graphics for the PC," in *Proc. of ACM SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series*, pp. 353–364, Aug. 1996. Article (CrossRef Link).
[9] A.C. Beers, M. Agrawala, and N. Chadda, "Rendering from Compressed Textures," in *Proc. of ACM SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series SIGGRAPH96*, pp. 373–378, Aug., 1996. Article (CrossRef Link).
[10] G. Knittel, A. Schilling, A. Kugler, and W. Strasser, "Hardware for Superior Texture Performance," *Computers & Graphics.* vol.20, no.4, pp. 475–481, July 1996. Article (CrossRef Link).
[11] Y. Xiao, C. S. Leung and P. M. Lam, "Self-organizing map-based color palette for high-dynamic range texture compression," *Neural Computing and Applications*, vol.21, pp. 639-647, June 2012. Article (CrossRef Link).
[12] C.C. Chen, X. Xu, R.L. Liao, W.H. Peng, S. Liu, and S. Lei, "Screen Content Coding using Non-square Intra Block Copy for HEVC," *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1-6, July 14-18, 2014. Article (CrossRef Link).
[13] D. Kwon and M. Budagavi, "Fast intra block copy (IntraBC) search for HEVC screen content coding," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 9-12, June 1-5, 2014. Article (CrossRef Link).
[14] K. Rapaka, C. Pang, J. Sole, M. Karczewicz, B. Li and J.Z. Xu, "Improved intra block copy and motion search methods for screen content coding," in *Proc. of SPIE, Applications of Digital Image Processing*, vol. 9599, pp.14 Sep. 2015. Article (CrossRef Link).
[15] Ericsson Labs, "Ericsson Texture Compression Tool etcpack v2.73:ETC1," Article (CrossRef Link).
[16] W. Griffin and M. Olano, "Evaluating Texture Compression Masking Effects Using Objective Image Quality Assessment Metrics," *IEEE Transactions on Visualization and Computer Graphics*, vol.21, no.8 , pp. 970-979, 2015. Article (CrossRef Link).

Li Cui received the B.S. degree from Yanbian University, Yanji, China in 2000 and the M.S. degree from Honam University, Korea in 2007. She is currently working toward a Ph.D. at Hanyang University, Seoul, Korea. And she is also a lecturer in the Department of Computer Science and Technology, College of Science and Engineering, Yanbian University, Yanji, China. Her current research interests include MPEG Reconfigurable Graphics Coding, Computer Graphics and Point Cloud Compression.

Euee S. Jang received a B.S. from Jeonbuk National University, Korea and a Ph. D. from SUNY at Buffalo, NY, USA. He is a Professor in the Dept. of Computer Science & Engineering, Hanyang University, Seoul, Korea. His research interests include image/video coding, reconfigurable video coding, and computer graphics objects. He has authored more than 150 MPEG contribution papers, more than 30 journal or conference papers, 35 pending or accepted patents, and two book chapters. Dr. Jang has received three ISO/IEC Certificates of Appreciation for contributions to MPEG-4 development. He received the Presidential Award from the Korean Government for his contribution to MPEG standardization.