ERROR ANALYSIS OF 3DC-BASED NORMAL MAP COMPRESSION AND ITS APPLICATION TO OPTIMIZED QUANTIZATION

Toshihiko Yamasaki and Kiyoharu Aizawa

Department of Information and Communication Engineering, The University of Tokyo

ABSTRACT

Normal mapping is one of the most essential technologies for realistic three-dimensional computer graphics. In conventional normal map compression such as 3Dc, only the *x* and *y* components are encoded and the *z* components are restored based on the normalizing condition. In this paper, we present an intuitively comprehensive error analysis for this approach. As a result, we reveal in what condition compression error becomes larger. We also present a non-linear quantization algorithm based on the formula for better compression performance than the conventional approaches. Experimental results using 300 normal map demonstrate that the PSNR is improved by 0.29 dB on average. Our algorithm is compatible with random access and highly-parallel processing on GPU.

Index Terms— Normal map, compression, error analysis, non-linear quantization, 3Dc

1. INTRODUCTION

Normal mapping [1], which is extended from bump mapping [2], is an essential technology in realistic three-dimensional (3D) computer graphics (CG). Normal mapping is a technology to express bumpy texture using simple polygons and extra texture data called normal maps. Normal maps are the maps of 3D normal vectors of the objects' surface stored in 2D arrays. Normal maps are often expressed as 24-bit full color bitmaps based on the following equation (thus, *x, y,* and *z* values are discrete and such data format are called 8x8x8 normal maps):

$$(x, y, z) = \frac{2}{255}(R, G, B) - 1 \qquad (1)$$

where (*x*, *y*, *z*) and (*R*, *G*, *B*) represent the element values of a normal map and their corresponding pixel values, respectively. The [-1, 1] range of normal vectors is mapped to integer values of [0, 255] based on (1). Also, the length of the normal vectors is normalized to one:

$$x^2 + y^2 + z^2 = 1. \qquad (2)$$

Here, the *z* component is always equal to or greater than zero because normal vectors point to the direction of the outer side of the surface:

$$-1 \le x \le +1,\ -1 \le y \le +1,\ 0 \le z \le +1 \qquad (3)$$

From (2) and (3), the *z* components can be eliminated since they can be restored by the simple calculation:

$$z = \sqrt{1 - x^2 - y^2}\ . \qquad (4)$$

Only by taking the inner product between the illumination vectors and the normal vectors, we can give more reality to the shading without using a lot of polygons. In addition, the shading can be calculated efficiently even if the light source position or the view point changes.

Since the bus bandwidth between CPU and GPU is quite limited, the data transfer of the high quality and large-seized normal maps is becoming a significant issue. Although normal maps can be expressed as regular bitmaps as discussed above, conventional 2D image compression algorithms such as JPEG or JPEG2000 are not applicable. This is due to the requirements peculiar to GPU-based processing such as random accessibility (i.e., fixed-rate compression) and the feasibility for the massively parallel processing. In this regard, there has been a number of dedicated normal map compression algorithms [3]-[12] developed so far though they are not always very efficient from the view point of compression ratio.

Ref. [3] was the first contribution to the normal map compression in which how much accuracy (resolution) is needed for human eyes instead of using 32-bit floating point accuracy was discussed. In [5], general-purpose texture compression algorithms using VQ was applied. ATI developed a dedicated algorithm called 3Dc [6] based on a block truncation algorithm [13]. This algorithm is now a *de-facto* standard supported by major GPU vendors. Later on, the 3Dc algorithm was improved by dynamically changing the bit allocation depending on the data distribution in the sub-blocks [7] and by the near-optimal principal component analysis [8]. Another modification of the 3Dc algorithm can be found in [9], in which all-zero sub-blocks are treaded in a different way for more efficient compression. A variable bit rate encoding approach [10] and hybrid method employing vector quantization (VQ) and Huffman encoding [11][12] were also presented so far. In many cases, the components are neglected based on (4). However, there is little work to investigate how the error in the *z* components is affected by the lossy compression of the *x* and *y* components.

In this point of view, the authors proposed an error model for the normal map compression algorithms [14]. However, the equation was difficult to understand intuitively.

In addition, how to apply the error model to better compression was not discussed in [14]. The authors confirmed that judging whether to include the z components or not as proposed in [14] does not work properly with the 3Dc-based algorithms.

Therefore, the purpose of this paper is to establish a more comprehensible error analysis model of the z component elimination and apply it to the better compression. In our analogy, we demonstrate that the error in the z components are the functions of not only the errors in the x and y values but the z value itself. Besides, we demonstrate a non-linear quantization algorithm based on our model, in which PSNR is enhanced by 0.29 dB on average as compared to 3Dc. This is also better than the e3Dc algorithm [7] (0.22 dB on average for our data). The algorithm can also be embedded into the other 3Dc-based approaches [7][8] for more efficient compression. In addition, the proposed non-linear quantization is compatible with random access and parallel computing on GPU.

2. ERROR MODEL AND COMPRESSION

Let us begin from [14] due to the limited length of the paper. Assume that the original values of the i-th x and y components values are x_i and y_i and the decoded values are x_i' and y_i'. The restored z component (z_i') obtained from (4), would become

$$\begin{aligned} z_i' &= \sqrt{1-\left(x_i'^2+y_i'^2\right)} \\ &= \sqrt{1-\left(x_i^2+y_i^2\right)-\left(x_i'^2+y_i'^2-x_i^2-y_i^2\right)} \\ &= \sqrt{1-\left(x_i^2+y_i^2\right)}\sqrt{1-\frac{x_i'^2+y_i'^2-x_i^2-y_i^2}{1-\left(x_i^2+y_i^2\right)}} \\ &= z_i + \sum_{n=1}^{\infty} C\left(\frac{1}{2},n\right)\left(-\frac{\left(x_i'^2+y_i'^2-x_i^2-y_i^2\right)^n}{\left(1-\left(x_i^2+y_i^2\right)\right)^{n-\frac{1}{2}}}\right) \end{aligned} \quad (5)$$

where C(1/2, n) represents the generalized binomial coefficients.

As can be observed from (5), z_i' is affected not only by the accuracy of the lossy compression in x_i and y_i, but also the original z component (z_i). In particular, it is expected that if the original z value is close to zero (i.e., the texture is more random and rough), the error would become very large. Since the normal map technology was originally developed to express the roughness of the surface, this is a significant problem.

For instance, in the 3Dc [6], the x and y planes are divided into 4x4 sub-blocks and the maximum and the minimum values are expressed with 8-bit accuracy. Then, the intermediate values are expressed with 3-bit (eight values) accuracy by a simple linear quantization. Therefore, the bit rates become 8 bpp (8x2+3x16 = 64 bits for each sub-block).

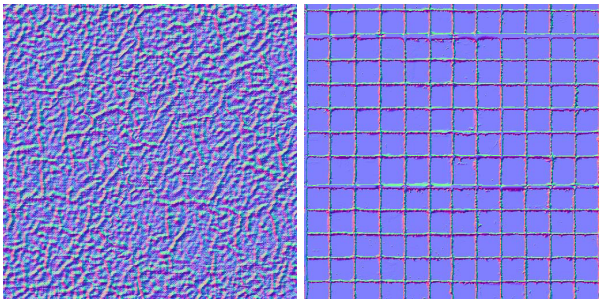

Fig. 1. Sample of normal maps: (a) rocky texture, (b) tile texture.

Although bit allocation is dynamically changed in some 3Dc-based approaches [7][8], quantization is still linear and the error characteristics as presented in (5) is not taken into consideration.

In this paper, we present a non-linear quantization approach based on the analysis in (5) by allocating more dense steps when |x| or |y| is close to one (e.g., near-zero region in the z components). Although our non-linear quantization would decrease the error in the z components, the errors in the x and y components would become larger instead. Therefore, the optimal encoding is in the trade-off between the error reduction in z and the error increase in x and y.

Since the normal vector distribution differs from normal maps to normal maps, it is very difficult to define a unique non-linear quantization steps. Therefore, we define several encoding modes and the encoder selects the best one by an exhaustive search. This would require a lot of encoding time (i.e., n times more encoding time for n types of encoding modes as compared to 3Dc). However, since encoding is carried out only once at the vendor side, the encoding complexity does not matter very much. Typically, the encoding time of the 3Dc algorithm is tens of milliseconds. Therefore, the encoding time for the proposed method is several seconds. In addition, we need to store the mode ID, but the additional bits for this information is only several bits and fixed rate, which is not a problem.

Possible variations in the encoding type are as follows:

- threshold value of $|x|_{max}$ and $|y|_{max}$ to apply our non-linear quantization
- how quantization steps are distorted.

Such information is common to all the sub-blocks in a normal map, thus saving the bits for encode mode memorization.

The decoding in our scheme is very close to that of 3Dc. The additional cost for our scheme is the detection of the encoding mode and simple arithmetic operations to decode the non-linearly quantized values. As a result, our algorithm is compatible with random access and parallel processing on GPUs.

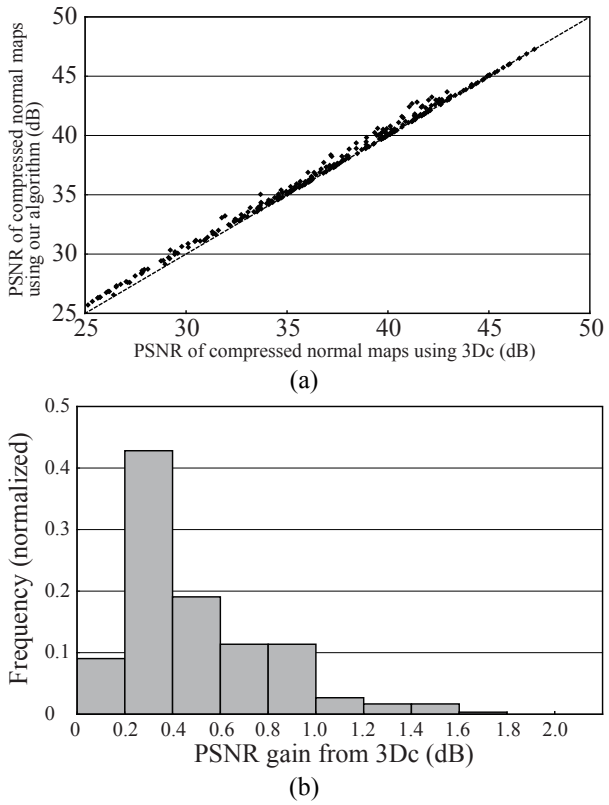

Fig. 2. PSNR improvement from original 3Dc: (a) distribution for 300 normal maps, (b) histogram of PSNR enhancement.

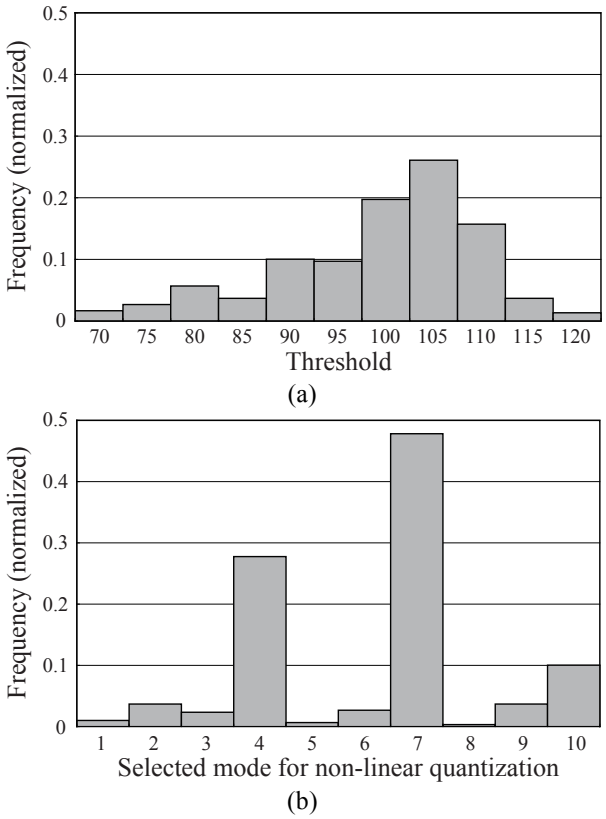

Fig. 3. Frequency histograms; (a) threshold, (b) mode for non-linear quantization.

3. EXPERIMENTAL RESULTS

The experiments were carried out using the normal maps in "Bump Texture Library" provided by Computer Graphics Systems Development Corporation [15]. About 300 normal maps were randomly selected for the experiments. The size of the normal maps were all 512x512. Two examples of the normal maps are shown in Fig. 1.

In order to demonstrate the effects of our non-linear quantization, the other parts in the encoding and decoding process were exactly the same as 3Dc [6]. The threshold value has 10 variations: {70/255, 75/255, 80/255, 85/255, 90/255, 95/255, 100/255, 105/255, 110/255, 115/255, 120/255}. The quantization mode is selected from one of the 10 kinds of steps listed below:

1. *min*, *min*+*step*, *min*+*step*x2, *min*+*step*x3, *max*-*step*x3, *max step*x2, *max*-*step*, *max*
2. *min*, *min*+*step*x0.7, *min*+ *step*x1.6, *min*+ *step*x2.6, *max*- *step*x2.6, *max*- *step*x1.6, *max*- *step*x0.7, *max*
3. *min*, *min*+ *step*x0.7, *min*+ *step*x1.6, *min*+ *step*x2.7, *max*- *step*x2.7, *max*- *step*x1.6, *max*- *step*x0.7, *max*
4. *min*, *min*+ *step*x0.7, *min*+ *step*x1.6, *min*+ *step*x2.8; *max*- *step*x2.8, *max*- *step*x1.6, *max*- *step*x0.7, *max*
5. *min*, *min*+ *step*x0.7, *min*+ *step*x1.7, *min*+ *step*x2.7, *max*- *step*x2.7, *max*- *step*x1.7, *max*- *step*x0.7, *max*
6. *min*, *min*+ *step*x0.7, *min*+ *step*x1.7, *min*+ *step*x2.8, *max*- *step*x2.8, *max*- *step*x1.7, *max*- *step*x0.7, *max*
7. *min*, *min*+ *step*x0.7, *min*+ *step*x1.7, *min*+ *step*x2.9, *max*- *step*x2.9, *max*- *step*x1.7, *max*- *step*x0.7, *max*
8. *min*, *min*+ *step*x0.7, *min*+ *step*x1.8, *min*+ *step*x2.8, *max*- *step*x2.8, *max*- *step*x1.8, *max*- *step*x0.7, *max*
9. *min*, *min*+ *step*x0.7, *min*+ *step*x1.8, *min*+ *step*x2.9, *max*- *step*x2.9, *max*- *step*x1.8, *max*- *step*x0.7, *max*
10. *min*, *min*+ *step*x0.7, *min*+ *step*x1.8, *min*+ *step*x3.0, *max*- *step*x3.0, *max*- *step*x1.8, *max*- *step*x0.7, *max*

where *min* and *max* represent the minimum and maximum values in each sub-block. And, *step* is defined as (*max*-*min*)/7. The mode #1 is the linear quantization employed in 3Dc, which is used for comparison. Therefore, the additional bit for a normal map is only 7 bits, which is almost negligible. These parameters were chosen by empirical study.

Fig. 2 shows how much PSNR is improved as compared to the original 3Dc at the same bit rates (8 bpp). Fig. 2(a) demonstrates the distribution of PSNR improvement. It is observed that our algorithm is valid regardless of the original PSNR obtained from the original 3Dc. Fig. 2(b) is a histogram of Fig. 2(a), showing how much PSNR is improved.

The PSNR is enhanced 0.29 dB on average. The PSNR improvement using e3Dc [7] was 0.22 dB for our data and that of the tight frame approach [8] was 0.02 dB. In [7][8], it was reported that e3Dc and the tight frame approach was 1.87 ~ 2.63 dB better than 3Dc. However, these algorithms work well in some limited cases, for instance, for artificial objects (buildings, etc.) or smoothly changing surfaces (car hoods, etc.). These algorithms are not suitable for random textures such as those for natural scenes (rocks, asphalt roads, etc.). On the other hand, our algorithm is preferred for more random and bumpy textures. Therefore, we can use both of the algorithms complimentary. Our algorithm can be easily implemented into [7][8].

Since the normal maps can be represented as RGB bit maps, we tend to compare the compression performance in terms of PSNR. However, normal maps are maps of normal vectors in which only the directions of the vectors are meaningful. Therefore, in [8], performance comparison by angle accuracy was also proposed. In this point of view, our algorithm yields the similar performance to the 3Dc algorithm. In both cases, the mean angle error was 3.1 degree.

Fig. 3 demonstrates the frequency histograms for the threshold values and the non-linear quantization modes. From Fig. 3(b), we can point out that our non-linear quantization is better than the linear quantization for the most cases (99%). The quantization mode #1 was chosen only three times. It is estimated that more parameter optimization is possible for both the thresholds and the quantization modes.

4. CONCLUSIONS

In this paper, we have presented an error model for the compressed normal maps when the z components are eliminated by taking advantage of the normalizing condition. It has been demonstrated that the error in the z values depends not only on the errors in the x and y values but also the original z values themselves. In addition, we have proposed a non-linear quantization for the 3Dc-based compression algorithms based on the error model. As a result, we have improved the PSNR by 0.29 dB as compared to the original 3Dc algorithm and by 0.07dB as compared to the more advanced algorithm (e3Dc). In addition, the non-linear quantization proposed in this paper is suited for the random access and the parallel computing in GPUs. In our future work, the dynamic thresholding and quantization mode selection in each sub-block is considered.

5. ACKNOWLEDGEMENTS

This work is supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under the "Development of fundamental software technologies for digital archives" project.

6. REFERENCES

[1] M. J. Kilgard, "A practical and robust bump-mapping technique for today's GPUs," *Game Developers Conference, Advanced OpenGL Game Development*, 2000.

[2] J.F. Blinn, "Simulation of wrinkled surfaces, Proc. the 5th annual conference on Computer graphics and interactive techniques," Vol. 12, No. 3, pp.286-292, 1978.

[3] M. Deering,. "Geometry compression," Proc. SIGGRAPH1995, pp. 13–20, 1995.

[4] S. Fenney and M. Butler, "Method and apparatus for compressed 3d unit vector storage and retrieval," Patent WO 2004/008394 A1, 2004.

[5] S. Green, "Bump Map Compression Whitepaper," http://download.nvidia.com/developer/Papers/2004/Bump_Map_Compression/Bump_Map_Compression.pdf, Oct. 2004.

[6] "ATI RADEON X800 3Dc white paper," www.ati.com/products/radeonx800/3DcWhitePaper.pdf.

[7] J. Munkberg, T.A. Möller, and J. Ström, "High-quality normal map compression", Proc. Graphics Hardware Workshop, pp. 95-101, 2006.

[8] J. Munkberg, O. Olsson, J. Ström, and T.A. Möller, "Tight frame normal map compression", Proc. 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware, pp. 37-40, 2007.

[9] B. Yang and Z. Pan, "A hybrid adaptive normal map texture compression algorithm," Proc. 16th International Conf. Artificial Reality and Telexistence (ICAT06), pp. 349-354, 2006.

[10] A. Wong and W. Bishop, "Adaptive normal map compression for 3d video games," Proc Future Play, 2006.

[11] T. Yamasaki and K. Aizawa, "Fast and efficient normal map compression based on vector quantization," Proc. IEEE ICASSP2006, pp. II-9-II-12, 2006.

[12] T. Yamasaki and K. Aizawa, "Highly efficient VQ-based normal map compression using quality estimation model," Proc. IEEE ICASSP2007, pp. I-1041-I-1044, 2007.

[13] E.J. Delp and O.R. Mitchell, "Image compression using block truncation coding," IEEE Trans. Communications, vol. com-27, no. 9, Sep. 1979.

[14] T. Yamasaki, K. Hayase, and K. Aizawa, "Mathematical error analysis of normal map compression based on unity condition," Proc. IEEE ICIP2005, pp. II-253-II-257, 2005.

[15] "Bump texture library," Computer Graphics Systems Development Corporation, http://cgsd.com/.