FAST AND EFFICIENT NORMAL MAP COMPRESSION BASED ON VECTOR QUANTIZATION

T. Yamasaki and K. Aizawa

Department of Frontier Informatics, Graduate School of Frontier Sciences
The University of Tokyo

ABSTRACT

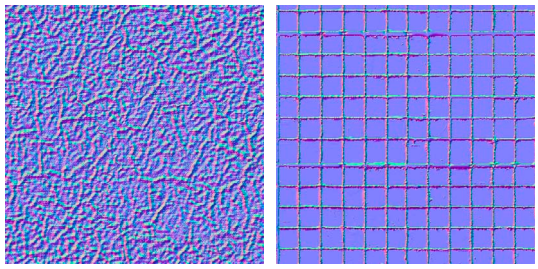
Normal maps play an important role in realistic 3D image rendering to express pseudo roughness of the surface with small amount of polygon data. In this paper, a fast and efficient normal map compression algorithm is proposed based on vector quantization and entropy coding. Using the strong correlation among *x*, *y*, and *z* components of normal maps owing to the unity condition, compression ratio has been made much better than conventional approaches. In addition, the encoding time has been made reasonable by considering the distribution of the data and employing inner product in nearest-neighbor search instead of Euclidian distance taking advantage of the unity condition of the training data.

1. INTRODUCTION

As more realism and visual complexity of scenes are required in 3D computer graphics, expressing 3D objects only by polygons has become problematic in terms of data storage and computational cost. In this regard, a number of elaborated texture mapping techniques have been developed such as bump mapping [1], displacement mapping [2], reflection mapping [3], relief mapping [4], and so forth. Bump mapping, which is nowadays extended to normal mapping [5] to express more detailed and complex texture, is especially a key technology to enhance the roughness and wrinkles of the surface with small amount of polygon data.

Although normal mapping was developed aiming at reducing the amount of polygon data, using high resolution normal maps for every surface of the whole of 3D objects cause a lack of video memory resources and a storage problem again. In this regard, some of the general-purpose texture compression algorithms have been applied to normal map compression [6]. In addition, a dedicated normal map compression algorithm called 3Dc has been implemented onto high-end Graphics Processing Units (GPU's) [7]. We have also been developing adaptive 3Dc algorithm [8] and efficient normal map compression algorithms using standard 2D image compression techniques such as JPEG and JPEG2000 [9][10].

However, such algorithms handle normal maps as 2D images and strong spatial correlation is assumed. Normal maps are usually more random and have less spatial correlation than 2D natural images, resulting in poor compression performance. For efficient compression, new data structure optimized for normal map data distribution and a dedicated compression algorithm is required.



(a) (b)

Fig. 1. Examples of normal maps: (a) wall texture; (b) tile texture. The size is 512x512.

The purpose of this paper is to develop a fast and efficient normal map compression algorithm using vector quantization (VQ) and Huffman encoding based on the strong correlation among *x*, *y*, and *z* components of normal maps owing to the unity condition. In this paper, each normal vector component (*x, y, z*) is handled as a training vector. Although the vector dimension is quite small, much better compression rate has been achieved. In addition, the encoding time has been made reasonable by considering the distribution of the data and employing inner product in nearest-neighbor search instead of Euclidian distance taking the unity condition of the training data into account.

2. NORMAL MAPS

Normal maps are the maps of three-dimensional vectors which represent directions of normal vectors of 3D object surfaces. Therefore, normal maps can be simply expressed as RGB bitmaps, in which the [-1, 1] range of normal vectors is mapped to [0, 255] based on the equation as described in the following (therefore, *xyz* values are discrete):

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

where (*x*, *y*, *z*) and (*R*, *G*, *B*) represent element values of each pixel in a normal map and their corresponding full color pixel values, respectively. Examples of normal maps are shown in Fig. 1.

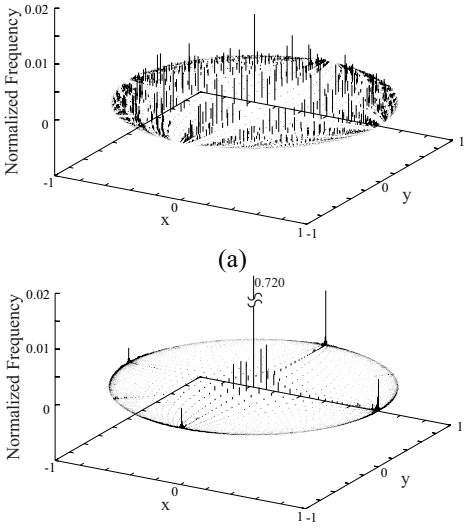


Fig. 2. Histograms of (*x, y, z*) vectors in normal maps: (a) wall texture; (b) tile texture.

In addition, the length of normal vectors is fixed at one in order to simplify the weight factor calculation of color and luminance into inner product operation between the normal vector and the luminance vector:

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

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

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

3. NORMAL MAP COMPRESSION

3.1. Vector Quantization Based on Data Distribution

In this paper, strong correlation among x, y, and z components are utilized. As described in Section 2, the length of normal vectors is normalized to one and therefore normal vectors are distributed on a unit sphere. In addition, z components are usually close to one. Therefore, distribution of normal vectors is quite limited and efficient compression will be achieved by VQ. The code indices are encoded by Huffman encoding for further compression. The codebook is designed for each normal map for better compression performance. Therefore, efficient training algorithm has been developed as described below for fast compression.

3.2. Training Efficiency Enhancement

Codebook training is computationally expensive in VQ algorithm. Therefore, the training efficiency is enhanced by the following techniques.

Distribution of normal vectors is quite limited and thus signal space of normal maps is very small. In addition, the dimension of the training vectors is only three. Therefore, a lot of identical vectors exist in training vectors. In Fig. 2, histograms of normal vector distribution are demonstrated. Frequency was normalized by the number of training vectors. In addition, z components are omitted since it can be restored using Eqs. (2) and (3). It is demonstrated that the number of unique vectors are small. For instance, the numbers of unique training vectors are 5776 and 15738 for Figs. 1(a) and 1(b), respectively, which is much smaller than the total number of training vectors: 262144 (512x512). Therefore, in our algorithm, unique vectors are firstly extracted and the number (i.e., frequency) of each unique vector is counted. The frequency table is utilized as weight factors for distortion calculation in the training phase. The detailed algorithms are shown as pseudo codes in Tables 1 and 2. It is shown that the number of *for* loops is drastically reduced and fast codebook generation is possible.

Due to the unity condition, nearest-neighbor code vector search for each training vector is equivalent to searching for such a code vector that maximizes the inner product with the training vector. This corresponds to the special case of gain-shape VQ [11]. The computational cost of inner production is smaller than that of Euclidean distance, thus enabling further speeding-up in training.

Table 1. Conventional training algorithm for VQ.

```
generate a training vector set from a M*N normal map
generate a seed code vector
while(codebook size <= desired size}){
 split the code vectors
 while(distortion is not minimum){
  for(i=0;i<M*N;i++){
   search for the nearest-neighbor (NN) code vector
   dist.=dist. + dist.(i-th training vector, NN code vector)
  }
  re-generate codebook
 }
}
```

Table 2. Proposed algorithm to decrease training time.

```
generate a training vector set from a M*N normal map
extract unique training vecs and generate the histograms of them
generate a seed code vector
while(codebook size <= desired}){
 split the code vectors
 while(distortion is not minimum){
  for(i=0;i<# of unique training vectors;i++){
   search for the nearest-neighbor (NN) code vector
   dist.=dist. + (freq. of the training vector) *dist.(i-th training
vector, NN code vector)
  }
  re-generate codebook
 }
}
```

3.3. Code Vector Update

Code vector updating is usually conducted by taking the average of all the training vectors in the same cluster. In normal map compression, the unity condition described in Eq. (2) needs to be considered:

$$H = \max_{x_m, y_m, z_m} \sum_{i \in S} \left(x_i \cdot x_m + y_i \cdot y_m + z_i \cdot z_m \right) \quad subject\ to\ x_m^2 + y_m^2 + z_m^2 = 1 \tag{4}$$

where S, (x_i, y_i, z_i), and (x_m, y_m, z_m) represent a set of vectors in a cluster, training vectors in a set S, and a code vector, respectively. By introducing Lagrangian method of undetermined multipliers, the code vector can be calculated as

$$(x_m, y_m, z_m) = \frac{1}{\sqrt{\left(\sum x_i\right)^2 + \left(\sum y_i\right)^2 + \left(\sum z_i\right)^2}} \cdot \left(\sum x_i, \sum y_i, \sum z_i\right) \tag{5}$$

This operation is equivalent to summing all the training vectors in a set S and normalizing the vector length to one.

4. EXPERIMENTAL RESULTS

In the experiments, a custom-made personal computer with Pentium 4 (3.2GHz) and 2GB memory was utilized. All the algorithms were implemented with Microsoft Visual C++ ver. 7.0 with *Ox* and *G7* options. In our VQ-based compression, LBG algorithm [12] was utilized with modifications in code vector training as explained in Section 3. The experiments were carried out using about 300 normal maps with the size of 512x512, which were randomly selected from "Bump Texture Library [13]."

Fig. 3 shows the average power of DCT coefficients of all the 8x8 sub-blocks using a standard 2D image (Lenna) and two kinds of normal maps. Luminance (Y) and x components were analyzed for Lenna and the normal maps, respectively. It is observed that more than ten times of power is contained in high-frequency region in normal maps, showing that conventional 2D image compression algorithms are not eligible for normal map compression.

Compression performance of our proposed algorithm and conventional approaches is demonstrated in Fig. 4. In the original 3Dc algorithm [7], scalar quantization based on block truncation coding algorithm [14] is employed for 4x4 blocks and the quantization level is fixed at eight (three bits). However, in the experiment, the quantization level was varied from two levels (one bit) to 16 levels (four bits) for comparison. In our proposed algorithm, codebook size and Huffman table are also included in the bit rate calculation. The quality of the compressed normal maps were evaluated by averaging mean square errors (MSE's) of x, y, z components and calculating peak signal noise ratio (PSNR). Strictly speaking, normal maps are not "images." Therefore, the quality of compressed normal maps needs to be analyzed by rendering 3D images using them. However, it has been demonstrated that the PSNR's of the compressed normal maps and those of the rendered images using them are almost the same [15]. As can be seen in Fig. 4, our algorithm is much better than the others in terms of compression ratio.

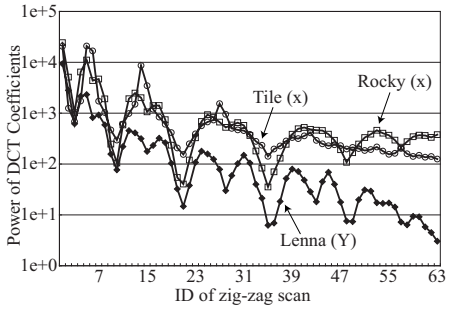


Fig. 3. DCT coefficients of standard image (Lenna) and normal maps.

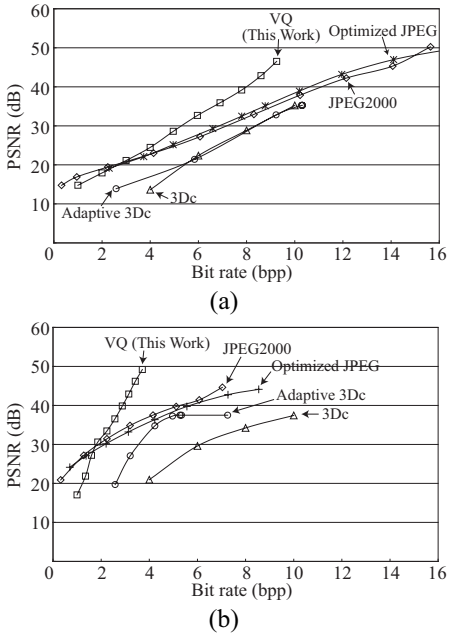
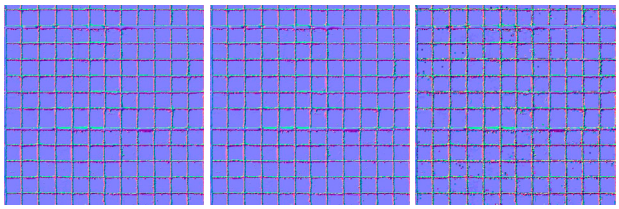


Fig. 4. PSNR's of compressed normal maps: (a) wall texture; (b) tile texture.

Our proposed algorithm yielded better compression ratio to more than 80% of the normal maps in the quality of 35dB and higher. Although it can be observed that the proposed VQ-scheme is outperformed by JPEG and JPEG 2000 in the very low bit-rate range, such low-bit normal maps below 35dB would result in significant noise and artifacts in the rendered images. Therefore, the proposed algorithm is eligible for practical usage.

The normal maps compressed at 4 bpp using various algorithms are shown in Fig. 5. It is observed that our algorithm gives much better visual quality than the 3Dc [7].

In Fig. 6, the average encoding/decoding time of the 300 normal maps is demonstrated. The encoding time for 64 codebook size was 4.8s for conventional VQ, 0.33s after introducing unique vector extraction, and 0.32s after introducing inner-product-based nearest neighbor search. For 1024 codebook size, the processing time was 52.6s, 1.19s, and 0.57s, respectively. It has been demonstrated that the larger the codebook size becomes and the more nearest neighbor search is required, the faster the proposed algo-



(a) (b) (c)

Fig. 5. Quality comparison of decoded normal maps of tile texture compressed approximately at 4 bpp: (a) This work; (b) optimized JPEG [10]; (c) 3Dc [7].

rithm performs. Decoding time was about 0.05s and almost constant for any codebook size.

Fig. 7 shows the encoding/decoding time comparison. Since the codebook needs be designed for each normal map, the training time is also contained in our method. The encoding time has been made reasonable and practicable as compared to the other algorithms. In addition, VQ-based algorithm is known to be very fast in decoding. This is an important advantage because decoding is conducted at the user side and executed much more times than encoding.

5. CONCLUSIONS

In this paper, an efficient compression technique for normal maps has been developed using vector quantization and Huffman encoding. While conventional normal map compression algorithms regarded normal maps as 2D images and to tried to compress them by reducing the two-dimensional spatial correlation, our approach is based on the strong correlation among (x, y, z) components of normal maps. Since most of normal map data are random and have little spatial correlation, our approach yielded much better compression performance than the conventional approaches. In addition, a fast codebook training algorithm for vector quantization has been developed by considering the distribution of the data and employing inner product in nearest-neighbor search instead of Euclidian distance taking advantage of the unity condition of the training data. As a result, the encoding time including the time for codebook training as well as the compression ratio have been made eligible for practical use.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] 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.

[2] J.D. Foley, A. Dam, S.K. Feiner, and J.F. Hughes, Computer Graphics PRINCIPLES AND PRACTICE, 2nd Edition, Addison-Wesley Publishing Company, 1996.

[3] J.F. Blinn, "Texture and reflection in computer generated images," Comm. ACM, Vol. 19, No. 10, pp. 542-547, 1976.

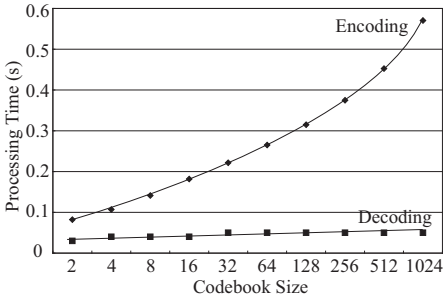


Fig. 6. Dependency of processing time on codebook size.

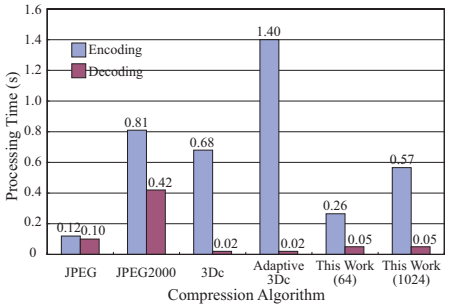


Fig. 7. Comparison of averaged encoding/decoding time.

[4] M.M. Oliveira, G. Bishop, D. McAllister, "Relief texture mapping," Proc. Siggraph2000, pp. 359-368, 2000.

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

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

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

[8] K. Hayase, T. Yamasaki, and K. Aizawa, "Compression of bump map for graphic object and proposal of its evaluation measure," Technical Report of IEICE, IE2004-174, pp.1-6, Sapporo, Feb. 2005 [In Japanese].

[9] R. Nakamura, T. Yamasaki, and K. Aizawa, "JPEG optimization for efficient bump map compression," Proc. of the 2005 IEICE General Conf., D-11-17, 2005 [In Japanese].

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

[11] M. J. Sabin and M. R. Gray, "Product code vector quantizers for waveform and voice coding," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, pp. 474–488, 1984.

[12] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," IEEE Trans. Communications, pp. 702-710, 1980.

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

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

[15] T. Yamasaki, K. Hayase, and K. Aizawa, "Mathematical PSNR prediction model between compressed normal maps and rendered 3D images," Proc. PCM 2005, Part II, LNCS 3768, pp. 584-594, 2005.