HIGHLY EFFICIENT VQ-BASED NORMAL MAP COMPRESSION USING QUALITY ESTIMATION MODEL

T. Yamasaki and K. Aizawa

Dept. of Information and Communication Engineering,
The University of Tokyo

ABSTRACT

Normal maps play an important role in computer 3D graphics to express pseudo roughness of the surface with a small amount of polygon data. In this paper, a highly efficient normal map compression algorithm is proposed based on an estimation model to predict the quality of the images rendered with the compressed normal maps. The optimal encoding is achieved by minimizing the predicted mean square error (MSE) employing vector quantization (VQ). In addition, encoding and decoding time is fast enough for practical usage. Experimental results demonstrate that the algorithm proposed in this paper yields better compression performance than the other algorithms in the literatures.

Index Terms— Normal map, normal mapping, computer graphics, compression, vector quantization

1. INTRODUCTION

Normal mapping, which is an extension of bump mapping [1], is a key technology for realistic 3D computer graphics because it can express pseudoroughness of the 3D objects' surface only with a small amount of polygon. Normal mapping has attractive advantages in rendering complexity and storage efficiency as compared to expressing the elaborated bumpy texture of 3D objects using a lot of polygons.

With normal mapping being used for every surface of the scenes, the data size of normal maps is getting problematic. Although normal maps can be represented as full color bitmap images, their spatial correlation among neighboring pixels is quite low [2], making it difficult to apply conventional 2D natural image compression algorithms such as JPEG and JPEG2000. Therefore, some techniques for normal map compression have been developed in recent years [2]-[6]. In [3], general-purpose texture compression algorithms using VQ was applied. ATI developed a dedicated algorithm called 3Dc [4] and implemented it on their graphic hardware. Later on, the 3Dc algorithm was improved by 3dB on average by optimizing the bit allocation depending on the data distribution [6]. In addition, a fast and efficient compression algorithm was developed in [2] employing VQ by taking the advantage of the limited spatial distribution of the normal vectors.

However, all the algorithms so far have tried to reduce the MSE between the original normal maps and the compressed ones. Strictly speaking, the compression performance should be evaluated by the images rendered with the compressed normal maps, not by the quality of the compressed normal maps themselves. For this purpose, an equation to express the relationship between the compressed normal maps and the rendered images is required. Otherwise, a strategy for the MSE optimization cannot be defined. Therefore, we have presented a model to estimate the quality of the images rendered with compressed normal maps [7]. The model made it possible to predict the quality of the images without actually rendering the images. In addition, the computational cost is much lower than calculating the MSE using actually rendered images.

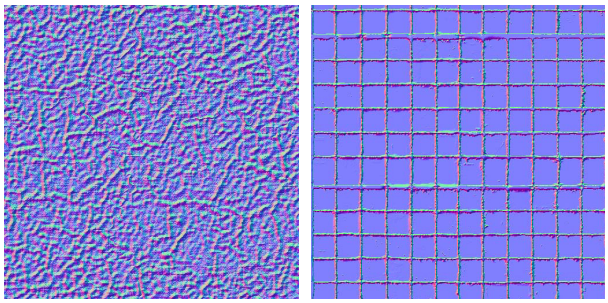
The purpose of this paper is to develop an efficient normal compression algorithm based on the quality estimation model [7]. Namely, normal maps are encoded considering the resultant images rendered with compressed normal maps. In our approach, a VQ-based compression algorithm is developed to minimize the predicted MSE. The encoding complexity has been reduced considering the distribution of normal vectors in normal maps. Experimental results demonstrate that the compression efficiency of our algorithm is as good or better than that of our previous work [2], which has given better performance than the other algorithms in most cases so far. In addition, both encoding and decoding time is fast enough for practical usage.

2. NORMAL MAP

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 integer values of [0, 255] based on (1) (therefore, *x, y,* and *z* values are discrete).

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

where (x, y, z) and (R, G, B) represent the element values of each pixel in a normal map and their corresponding full



(a) (b)

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

color pixel values, respectively. Examples of normal maps are shown in Fig. 1.

The length of normal vectors is normalized to one in order to simplify the weight factor calculation of color and luminance into an inner product between the normal vector and the luminance vector (see (4)):

$$x^2 + y^2 + z^2 = 1 \tag{2}$$

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

$$-1 \le x \le +1,\ -1 \le y \le +1,\ 0 \le z \le +1 \tag{3}$$

3. QUALITY ESTIMATION MODEL

In the quality estimation model [7], a simple but essential shading model is assumed. That is, there is no ambient light, light emission, attenuation/spotlight effects, nor specular. In addition, it is assumed that there is a white diffuse point light source in the infinite distance in the scene. Normal maps are mapped to a square board along with color texture data. The viewpoint is set at the right top of the board. Even when the normal maps are rendered on an object with complicated shape, the local area of the surface can be approximated as a square plane. Under this condition, the shading equation for each pixel is described as

$$\mathbf{I} = \max\left(\mathbf{L}\cdot\mathbf{N}\right)\mathbf{D} \tag{4}$$

where **I**, **L**, **N**=(x, y, z), and **D** represent the resultant pixel value in the rendered image, the luminance vector, the normal vector, and the color texture, respectively. For simplicity, the max function in (4) is neglected:

$$\mathbf{I} = \left(\mathbf{L}\cdot\mathbf{N}\right)\mathbf{D} \tag{5}$$

In the same manner, the resultant pixel value (**I**') using a compressed normal vector (**N**'=(x', y', z')) is described as

$$\mathbf{I}' = \left(\mathbf{L}\cdot\mathbf{N}'\right)\mathbf{D} \tag{6}$$

Here, we define the MSE between **I** and **I**' as

$$MSE = \frac{1}{3M}\sum_{i}^{M}\left\|\mathbf{L}_i\cdot\mathbf{N}_i - \mathbf{L}_i\cdot\mathbf{N}_i'\right\|^2\left\|\mathbf{D}_i\right\|^2 \tag{7}$$

where M is the number of pixels in the rendered image. Then, **L** and **N**-**N**' are defined as follows:

$$\mathbf{L} = \left(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta\right) \quad \left(0 \le \theta \le \pi/2, 0 \le \phi < 2\pi\right) \tag{8}$$

$$\mathbf{N} - \mathbf{N}' = \left(x - x', y - y', z - z'\right) = \left(\Delta x, \Delta y, \Delta z\right) \tag{9}$$

The averaged MSE for all possible light source location is calculated analytically as in (10).

$$\begin{aligned}\overline{MSE} &= \frac{1}{3M}\sum_{i}^{M}\frac{1}{2\pi}\int\!\!\int_{S}\left(\mathbf{L}_i\cdot\mathbf{N}_i - \mathbf{L}_i\cdot\mathbf{N}_i'\right)^2\left\|\mathbf{D}_i\right\|^2 dS \\ &= \frac{1}{6\pi M}\sum_{i}^{M}\int_0^{\frac{\pi}{2}}\int_0^{2\pi}\left(\mathbf{L}_i\cdot\mathbf{N}_i - \mathbf{L}_i\cdot\mathbf{N}_i'\right)^2\left\|\mathbf{D}_i\right\|^2 d\phi\sin\theta d\theta \\ &= \frac{1}{9M}\sum_{i}^{M}\left(\left(\Delta x_i\right)^2 + \left(\Delta y_i\right)^2 + \left(\Delta z_i\right)^2\right)\left\|\mathbf{D}_i\right\|^2\end{aligned} \tag{10}$$

As a result, the predicted peak signal to noise ratio ($PSNR_{model}$) of the rendered image is described as in (11).

$$\begin{aligned}PSNR_{\text{model}} &= 10\log_{10}\frac{255^2}{\overline{MSE}} \\ &= 10\log_{10}\frac{255^2\cdot 9M}{\sum_{i}^{M}\left(\left(\Delta x_i\right)^2 + \left(\Delta y_i\right)^2 + \left(\Delta z_i\right)^2\right)\left\|\mathbf{D}_i\right\|^2}\end{aligned} \tag{11}$$

The intuitive understanding of this equation is that the error in the compressed normal maps becomes invisible when the color texture is dark. On the other hand, when the color texture is bright, the error is amplified and degrades the resultant image. Please refer to [7] for the validity of this model.

4. COMPRESSION ALGORITHM

4.1. MSE minimization

Since the MSE of the image rendered with the compressed normal maps is predicted as in (10), our compression strategy is to minimize it using VQ. In our approach, a normal vector in each pixel of normal map is utilized as a vector because (10) is a function of (x, y, z). Although the vector dimension is only three, it will be demonstrated that the compression efficiency is rather high.

The code vectors (representative vectors in the clusters) should satisfy the following condition:

$$H = \min_{x_r, y_r, z_r}\sum_{i\in S}\left(\left(x_i - x_r\right)^2 + \left(y_i - y_r\right)^2 + \left(z_i - z_r\right)^2\right)\left\|\mathbf{D}_i\right\|^2 \quad \textit{subject to } x_r^2 + y_r^2 + z_r^2 = 1 \tag{12}$$

where S, (x_i, y_i, z_i), and (x_r, y_r, z_r) represent the set of vectors in a certain cluster, the i-th training vector in a set S, and the code vector, respectively. By the Lagrange multipliers, the code vector becomes

$$\left(x_r, y_r, z_r\right) = \frac{\left(X, Y, Z\right)}{\sqrt{X^2 + Y^2 + Z^2}} \tag{13}$$

where

$$\left(X, Y, Z\right) = \left(\sum_{i\in S}\left\|\mathbf{D}_i\right\|^2 x_i, \sum_{i\in S}\left\|\mathbf{D}_i\right\|^2 y_i, \sum_{i\in S}\left\|\mathbf{D}_i\right\|^2 z_i\right) \tag{14}$$

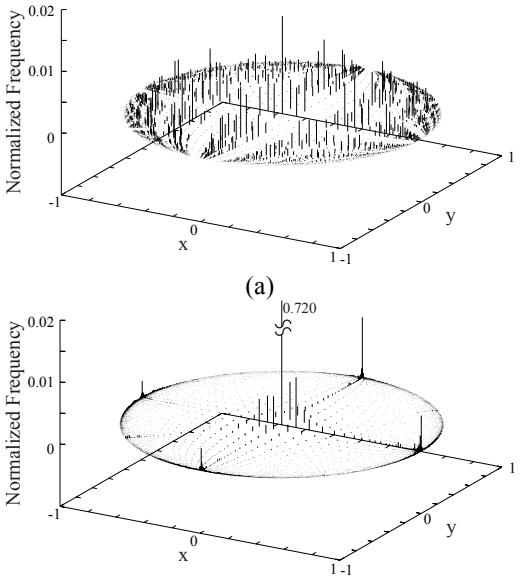


Fig. 2. Histograms of (*x, y, z*) vectors in normal maps: (a) wall texture; (b) tile texture.

4.2. Encoding time reduction

In VQ, codebook training is computationally very expensive because a large number of repetitions of distance calculation among vectors are required. Therefore, the training time is decreased by reducing the number of distance calculation as well as the cost for distance calculation.

Firstly, the spatial distribution of normal vectors is quite limited and thus the signal space of normal maps is very small. In addition, the dimension of the vectors is only three. Therefore, a lot of identical vectors exist in training vectors. Fig. 2 shows the histograms of the normal vector distribution. The frequency is normalized by the number of training vectors. In addition, the *z* components are omitted since it can be restored using (2) and (3). It is demonstrated that the number of the unique vectors are quite small. For instance, the numbers of unique training vectors are 5,776 and 15,738 for Figs. 1(a) and 1(b), respectively, which is much smaller than the total number of the training vectors: 262,144 (512×512). Therefore, in our algorithm, the unique vectors are extracted in advance and the nearest-neighbor search is conducted only for them. At the same time, $\Sigma\|\mathbf{D}\|^2$ is calculated for each unique vector. This value is used for the distortion calculation and the code vector generation.

Since the length of normal vectors is normalized to one as discussed in Section 2, searching the nearest neighbor (NN) vector is equal to searching the vector that gives the maximum inner product. The cost for the inner product is smaller than that for the square error calculation, thus further reducing computational cost. The detailed algorithm is shown as a pseudo code in Table 1.

Table 1. Quasi code for our normal map compression.

```
1.  generate a training vector set from a normal map
2.  extract the unique training vectors
3.  calculate Σ||D||^2 for each unique vector
4.  generate a seed code vector using (13) and (14)
5.  while(codebook size <= desired}){
6.    split the code vectors
7.    while (distortion is not minimum){
8.      for (i=0; i<# of unique training vectors; i++){
9.        search for the NN code vector using the inner product
10.       distortion=distortion + (Σ||D||^2 of the i-th unique vector)
      *distortion(the i-th unique vector, its NN code vector)
11.     }
12.   regenerate codebook using (13) and (14)
13.   }
14. }
15. encode list of unique vectors and code indices
```

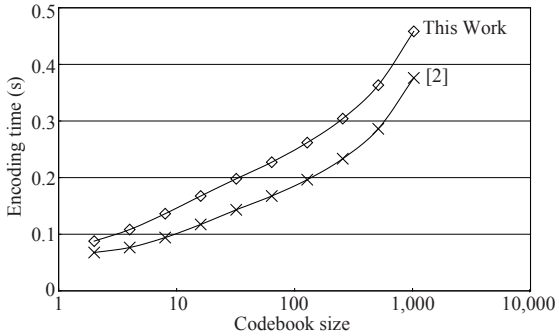
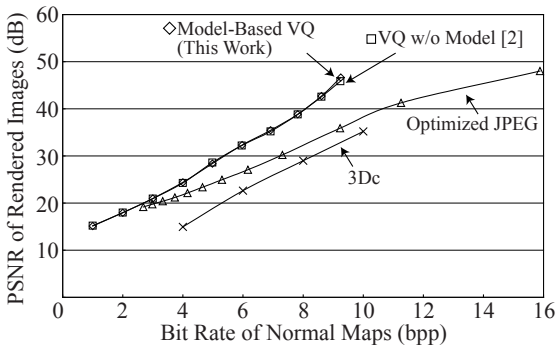


Fig. 3. Encoding time as a function of codebook size.

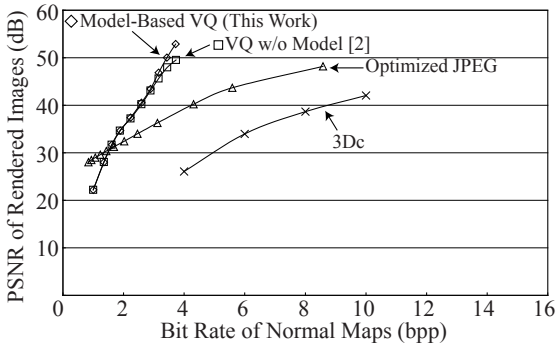
5. EXPERIMENTAL RESULTS

In the experiments, a personal computer with Pentium 4 (3.2GHz) and 2GB memory was utilized. The algorithm was implemented using gcc 3.4.4. The experiments were carried out using about 300 normal maps with the size of 512×512 that were contained in "Bump Texture Library [8]." In the experiments, the light source was changed according to the following conditions: θ=(0, 15, 30, 45, 60, 75), ϕ=(0, 45, 90, 135, 180, 225, 270, 315). Then, the mean PSNR considering all the light positions was calculated. The other rendering conditions were the same as those in Section 3. The performance was compared with that in [2], which has yielded the best performance in most normal maps among the previous works [2]-[4][6].

The mean encoding time as a function of codebook size is shown in Fig. 3. Although a certain amount of overhead time (~0.1 s) is consumed for calculating (14) as compared to [2], the encoding time is less than 0.5 s. Since decoding does not require any additional processing, the decoding time was about 50 ms for any codebook size, which is the same as that in [2]. This is an important advantage because decoding is conducted by the user device where computational power is typically not as great.



(a)



(b)

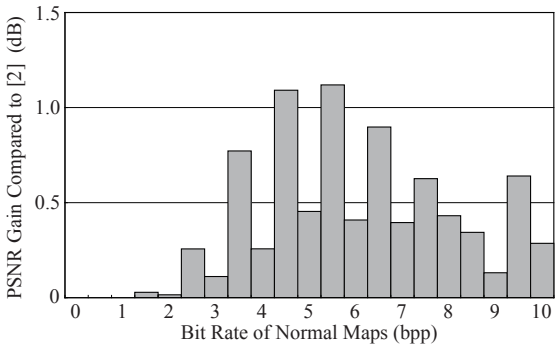
Fig. 4. PSNR of the images rendered with compressed normal maps: (a) wall texture; (b) tile texture.

In Fig. 4, the actual PSNR (not $PSNR_{model}$) of the rendered images using the compressed normal maps in Fig. 1 is illustrated. It is shown that the algorithm proposed in this paper yields as good or better compression performance than [2]. For instance, PSNR is enhanced by 0.8dB in Fig. 4(a) and 4.0 dB in Fig. 4(b) at the high PSNR region. It was observed that the PSNR is enhanced considerably in higher PSNR region: 35dB and above. Since the normal mapping was developed aiming at high-quality image rendering, the quality enhancement in such a high PSNR region is important. It is also shown that our algorithm is much better than other approaches [4][5].

Fig. 5 shows how much the PSNR is improved on average for all the normal maps as compared to that in [2]. The PSNR gain is demonstrated as a function of bit rates. It can be seen that the PSNR is increased by 0~1.1 dB.

6. CONCLUSIONS

In this paper, an efficient normal compression algorithm was presented using the quality estimation model. In our approach, the encoding was done considering the quality of the images rendered with compressed normal maps. As a result, the compression performance was enhanced by 0~1.1 dB as compared to our previous work with a small computational time overhead of 0.1 s.



(b)

Fig. 5. Mean PSNR gain for about 300 normal maps as compared to [2] as a function of bit rate.

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.

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

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

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

[5] 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.

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

[7] T. Yamasaki, K. Hayase, and K. Aizawa, "Mathematical PSNR prediction model between compressed normal maps and rendered 3D images," Proc. 2005 Pacific-Rim Conference on Multimedia (PCM 2005), LNCS 3768, pp. 584-594, 2005.

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