Texture Coordinate Compression for 3-D Mesh Models Using Texture Image Rearrangement

Sung-Yeol Kim, Young-Suk Yoon, Seung-Man Kim,
Kwan-Heng Lee, and Yo-Sung Ho

Gwangju Institute of Science and Technology (GIST),
1 Oryong-dong, Buk-gu, Gwangju 500-712, Korea
{sykim75, ysyoon, sman, leee, hoyo}@gist.ac.kr

Abstract. Previous works related to texture coordinate coding of the three-dimensional(3-D) mesh models employed the same predictor as the geometry coder. However, discontinuities in the texture coordinates cause unreasonable prediction. Especially, discontinuities become more serious for the 3-D mesh model with a non-atlas texture image. In this paper, we propose a new coding scheme to remove discontinuities in the texture coordinates by reallocating texture segments according to a coding order. Experiment results show that the proposed coding scheme outperforms the MPEG-4 3DMC standard in terms of compression efficiency. The proposed scheme not only overcome the discontinuity problem by regenerating a texture image, but also improve coding efficiency of texture coordinate compression.

Keywords: 3-D mesh coding, texture coordinate compression, texture image rearrangement.

1 Introduction

As high-speed networks and the Internet are widely used, various multimedia services with three-dimensional(3-D) audio-visual data have been proposed, such as a realistic broadcasting, immersive 3-D games, and 3-D education tools. These multimedia applications require not only high quality visual services, but also user-friendly interactions.

The 3-D mesh model, which represents 3-D objects by geometry, connectivity, and photometry information, is popular as one of the standard representations of 3-D objects. The geometry information describes 3-D coordinates of vertices, and the connectivity information describes the topology with the incidence relations among vertices, edges and faces. The photometry information includes surface normal vectors, colors, and texture coordinates, which are the attributes of vertices needed to render the 3-D mesh model.

Various processing techniques for the 3-D mesh model have been proposed. The mesh deformation method[1] changes the 3-D mesh model into arbitrary shapes, and the mesh refinement method[2] makes 3-D surfaces more smoothly using subdivision algorithms. In addition, the mesh simplification method[3,4] is

Y.-S. Ho and H.J. Kim (Eds.): PCM 2005, Part I, LNCS 3767, pp. 687–697, 2005.

widely used to support the level-of-detail(LOD) in the 3-D scene and to transmit the 3-D mesh model data progressively. The 3-D mesh compression method[5] is one of the important techniques in the 3-D mesh processing. In general, the representation of 3-D mesh models requires a tremendous amount of data. In order to store the 3-D mesh data compactly and transmit them efficiently, we need to develop efficient coding schemes for the 3-D mesh model. Figure 1 lists the major approaches for 3-D mesh compression.

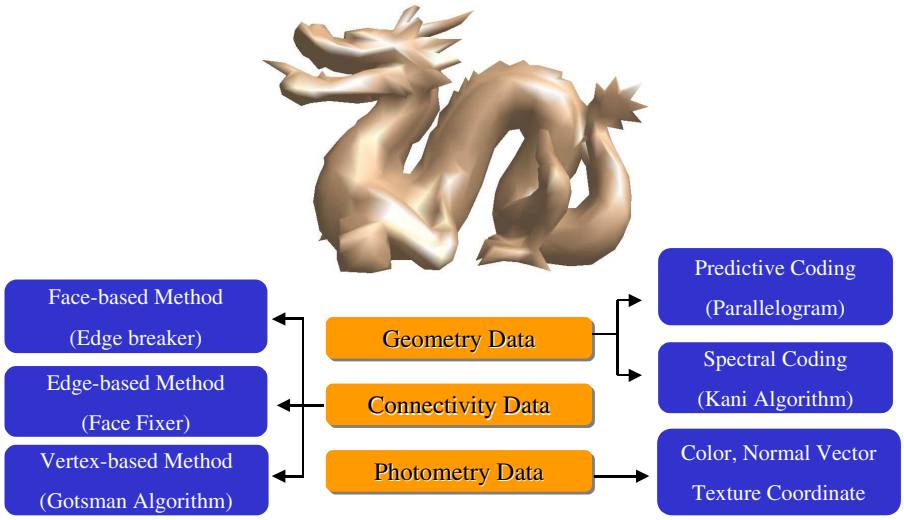


Fig. 1. Approaches for 3-D mesh compression

Previous 3-D mesh coding schemes have focused on compressing geometry and connectivity information. However, we should note that photometry information contribute around 60% to the entire 3-D model data. Especially, when we acquire a 3-D model data from a 3-D data acquisition device, such as a 3-D scanner, we usually employ a texture mapping technique with texture coordinates and texture images instead of colors. The texture mapping makes the acquired models more realistic. Moreover, since the texture coordinate data are corresponding to around 40% of the photometry data, it is necessary for us to develop an efficient texture coordinate coding technique. In this paper, we focus on compressing the texture coordinates.

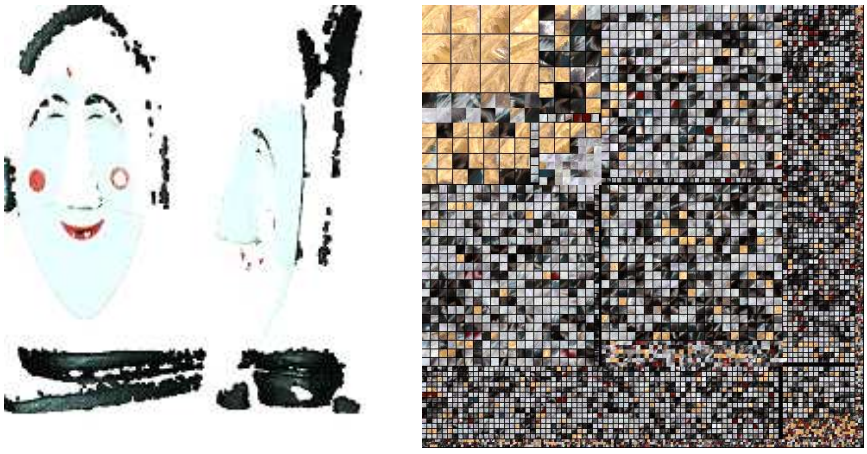
This paper is organized as follows. Section 2 explains previous works, and Section 3 describes the proposed texture coordinate coding method. After we provide experimental results in Section 4, we conclude in Section 5.

2 Texture Coordinate Coding Algorithms

Texture mapping techniques are widely used to increase the realism of 3-D mesh models. Texture images and texture coordinates are needed to render a 3-D mesh

model using a texture mapping technique. The 2-D texture images are mapped into every polygon of a 3-D mesh model according to its texture coordinates.

There are two types of texture images: atlas and non-atlas texture images. Atlas texture images, which assemble the segmented charts into a single texture image, are sometimes convenient to match with 3-D mesh models. Non-atlas texture images are the set of little squares, that are parameterized into a single texture. Figure 2 compares the atlas and non-atlas texture images.



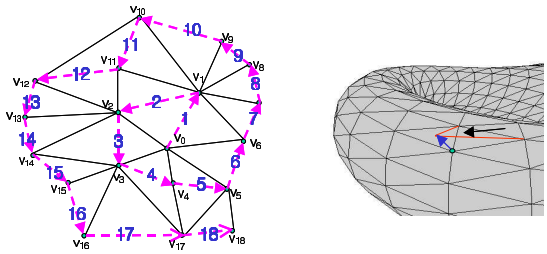
(a) Atlas texture image (b) Non-atlas texture image

Fig. 2. Atlas texture image vs. non-atlas texture image

In general, texture images are commonly coded by the JPEG standard, as still-frame images. According to the quantization design, coding efficiency and quality for texture images will be determined. In this paper, we will not consider the coding for texture images, but the coding for the texture coordinate data. In order to compress texture coordinates, previous works usually exploited the same predictor that was used to code the geometry data. Although there were some efforts to code the texture coordinates by different schemes, they did not increase coding efficiency significantly.

Deering[6] and Taubin *et al.*[7] traversed all the vertices according to the connectivity information, and coded texture coordinates using a linear predictor. Similarly, Touma and Gotsman[8] encoded the topology information by traversing the vertices, and texture coordinates were coded by predicting them along the traversal order using a parallelogram predictor as shown in Fig. 3. The predicted errors of the texture coordinates were entropy-coded. Isenburg[9] introduced discontinuity problem between the texture images and the texture coordinates during texture coordinate coding, and they proposed a texture coordinate coding scheme with a selective linear predictor.

Although previous texture coordinate algorithms yielded good performance on the 3-D mesh models with atlas texture images, they had problems



(a) Vertex traversal (b) Parallelogram Predictor

Fig. 3. Vertex traversal and predictive coder

for 3-D models with non-atlas texture images because of discontinuity problem. For realistic multimedia applications, we need to obtain 3-D mesh models from 3-D data acquisition devices to increase the realism. In general, when we obtain 3-D mesh models from a 3-D data acquisition device, most 3-D mesh models include non-atlas texture images. Therefore, we should develop a good coding scheme for texture coordinates with non-atlas texture images.

2.1 Discontinuity in the Texture Coordinates

Texture mapping is a procedure that maps a 2-D texture image into each polygon of the 3-D mesh model. Although each polygon can be mapped independently, it is beneficial to map neighboring polygons into neighboring texture images. We call the process for obtaining suitable texture coordinates as texture parametrization. The texture coordinates are generated through the texture parametrization between vertex coordinates and texture images. The 3-D mesh models with non-atlas texture images cause the texture parametrization to be broken.

When we code the texture coordinates with non-atlas texture images along a vertex traversal order, the coded texture coordinates will be sorted out in an unreasonable order, called as discontinuity in texture coordinates. The predicted

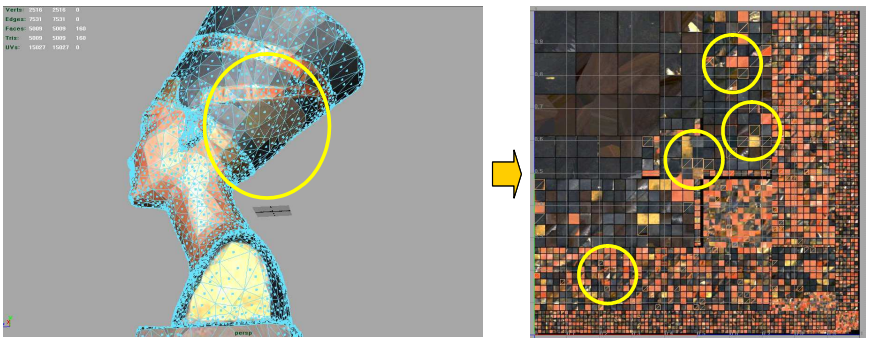


Fig. 4. Discontinuity in texture coordinates

errors of texture coordinates will be increased when we compress the texture coordinates by a predictive coding scheme owing to the discontinuities. As a result, discontinuities in the texture coordinates deteriorate coding efficiency drastically. Figure 4 shows discontinuities in the texture coordinates.

3 Texture Coordinate Coding with Texture Image Rearrangement

In this paper, we propose a new predictive coding scheme for texture coordinate compression using a texture image rearrangement. The main contribution of our proposed scheme is that we remove the discontinuity in the texture coordinates before employing a predictive coder to compress texture coordinates. The proposed scheme can be divided into four parts: analysis of the mesh model, analysis of the texture coordinate, rearrangement of the texture image, and predictive coding of the texture coordinate, as shown in Fig. 5.

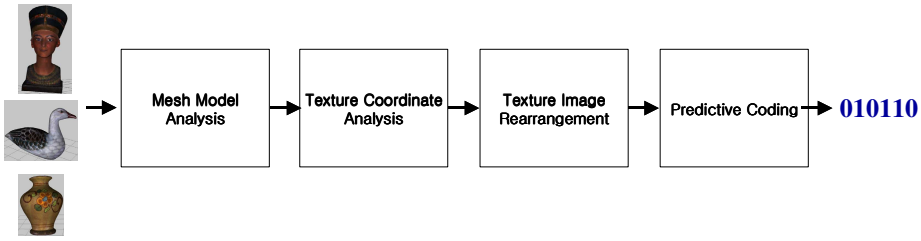


Fig. 5. Overall flow for texture coordinate compression

3.1 Analysis of the Mesh Model

Analysis of the mesh model is a process to extract a texture image and texture coordinates from the 3-D mesh model. The input data format of 3-D mesh models are usually the virtual reality modelling language(VRML)[10]. In order to analyze the 3-D mesh model, we need to obtain a texture image and texture coordinates through a VRML analyzer. From a part of the VRML file to represent the Nefertiti model, we can notice that the Nefertiti model includes a texture image and texture coordinates.

```
texture ImageTexture {
        url "nefert5_T5k1.bmp"
      }
texCoord TextureCoordinate { point [
0.333008 1.13184, 0.333008 1.11621, 0.348633 1.13184,
0.192383 1.19434, 0.192383 1.20996, 0.176758 1.19434,
0.750977 1.53027,
```

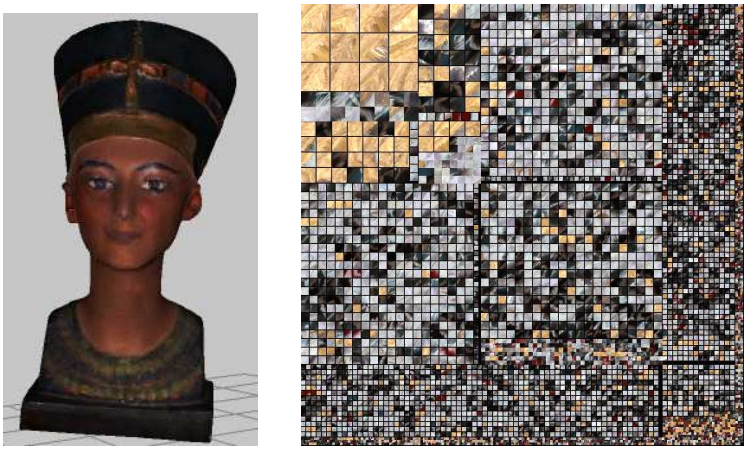


Fig. 6. Example of 3-D mesh model and its texture image

The texture image of the Nefertiti model is $nefer5_t5k1.bmp$, which has 14993 texture coordinates. In general, texture coordinates have x coordinates and y coordinates since they indicate texture positions in a 2-D texture image. With the VRML analyzer, we also extract the geometry information, the connectivity information and other photometry data, such as colors and normal vectors. In order to compress the texture coordinate data, we traverse all vertices to get a coding order using the extracted geometry and connectivity data. Figure 6 describes the Nefertiti model and its texture image.

3.2 Analysis of the Texture Coordinate

In general, texture coordinates are normalized with values between 0 and 1. In order to analyze the texture coordinate, we need to obtain exact texture positions in the texture image from normalized texture coordinates. Texture positions are obtained by multiplying the width and the height of the texture image to texture coordinates as described by Eq.1 and Eq.2.

$$x_{pos} = TextureWidth \times x_{norm} \tag{1}$$

$$y_{pos} = TextureHeight \times y_{norm} \tag{2}$$

After obtaining texture positions, we put the texture image and the corresponding texture positions into a memory. We may have an unpredictable problem in extraction of texture positions since the axis of the texture image can be inverse. Therefore, we should make sure that the axis of the texture image should not be inverse.

During the analysis of the texture coordinate, we search for the discontinuous points between a texture image and texture coordinates. In order to obtain the discontinuous points, we first define a searching order. In this paper, we use a vertex traversal algorithm[11,12] supported by 3-D mesh coding(3DMC)

in MPEG-4 standards[13] to obtain a searching order. Before searching for discontinuous points, we should define a threshold to determine whether a texture coordinate is discontinuous or not. When distances between a texture coordinate and neighboring texture coordinates are larger than the threshold value, we regard the texture coordinate as a discontinuous point. Finally, discontinuous points are stored in a memory using a data structure.

3.3 Rearrangement of the Texture Image

By the rearrangement of the texture image, we reallocate the texture segments continuously with the discontinuity information. Texture segments are rearranged by the zig-zag order along a coding order. Before rearranging texture segments, we allocate a image buffer so as to store a regenerated texture image.

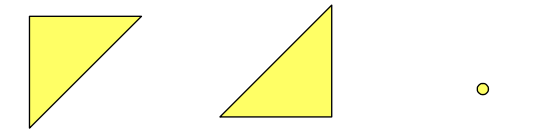


Fig. 7. Types of texture segments

In rearrangement of the texture image, we extract the texture segments corresponding to a triangle face from the texture image. After checking the data structure including the discontinuous information, we allocate texture segments into the image buffer continuously. There are three types of texture segments in non-atlas texture images as shown in Fig. 7. One has a triangle shape with lower direction. Another is a triangle shape with upper direction. The other is a point, which indicates that three texture coordinates are all the same. In order to generate a rearranged texture image, we first classify the type of texture segments. Then, we allocate texture segments into a image buffer by moving the buffer position.

The texture coordinates of a texture segment are changed according to the resolution of a regenerated texture image. In order to use the discontinuous points in a predictive coder, we indicate a flag for each discontinuous points. The flags are used for deceasing the residual errors in the predictive coder. Figure 8 shows an example of texture image rearrangement.

After rearrangement of texture image, we regenerate the texture coordinates corresponding to the reallocated texture image. As soon as we reallocate the texture segments, we calculate the texture coordinates. Finally, we normalize the texture coordinate data by dividing them with the width and height of the regenerated texture image using Eq. 3 and Eq. 4.

$$x_{norm} = x_{pos} \div NewTexutureWidth \tag{3}$$

$$y_{norm} = y_{pos} \div NewTexutureHeight \tag{4}$$

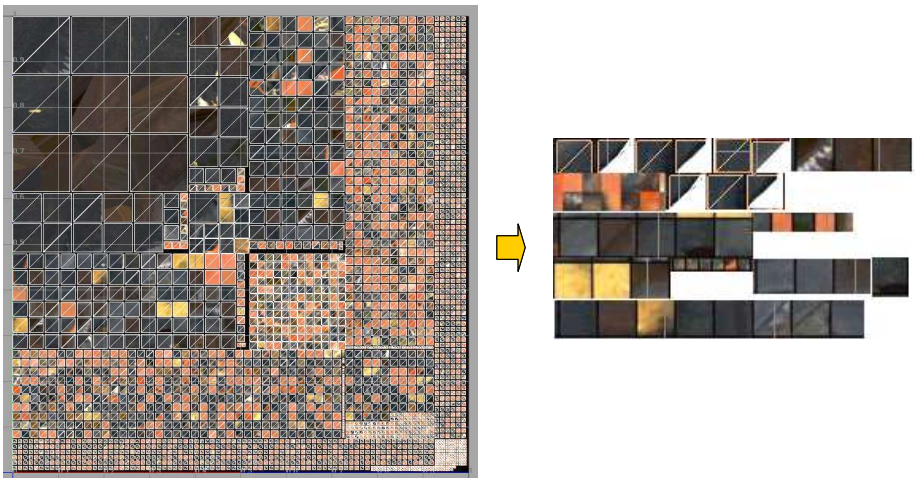


Fig. 8. Texture image rearrangement

3.4 Predictive Coding of the Texture Coordinate

For the predictive coding of texture coordinates, we obtain residual errors using MPEG-4 3DMC. The texture coordinates are coded through the parallelogram predictor along a vertex traversal order. Figure 9 shows the parallelogram predictor. In order to predict a vertex (tx_4, ty_4), we obtain a referred vertex $(tx_4, ty_4)'$ from Eq. 5.

$$(tx_4, ty_4)' = (tx_1, ty_1) + (tx_2, ty_2) - (tx_3, ty_3) \tag{5}$$

From Eq. 6, we can obtain the residual error (ex_4, ey_4). Finally, the residual errors (ex_4, ey_4) are coded through a variable length coder.

$$(ex_4, ey_4) = (tx_4, ty_4) - (tx_3, ty_3) \tag{6}$$

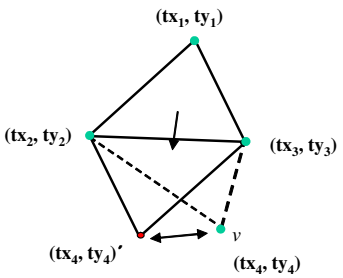


Fig. 9. Predictive coder

4 Experimental Results and Analysis

We have evaluated the proposed algorithm with the Coin model and the Nefertiti model. The Coin model consists of 1468 vertices, 2932 faces and a texture image with 512 x 512 resolutions. The Nefertiti model has 2501 vertices, 4998 faces and a texture image with 515 x 512 resolutions. The Coin model has 8796 texture coordinates and The Nefertiti model has 14993 texture coordinates. Figure 10 shows the tested models and their texture images.

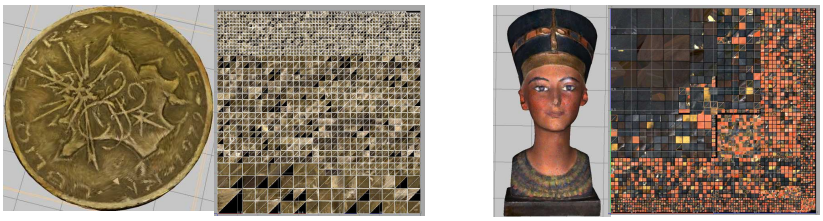


Fig. 10. Tested models and texture images

In order to compress the texture images, we used the JPEG with the quantization table suggested in the standard. Whereas, we used the 3DMC reference software implemented by the MPEG-4 standard for compressing the texture coordinate data. We experimented our proposed scheme for the tested models with a Pentium-4 personal computer including 512 MB memory and a Window operation system.

Table 1. Texture coordinates comparison between 3DMC and the proposed scheme

	3DMC		proposed scheme	
	variance of residuals	predictive coding	variance of residuals	predictive coding
Coin	25.44	168.4 KB	13.22	106.3 KB
Nefertiti	27.64	287.5 KB	14.27	184.3 KB

Table 2. Texture image comparison between 3DMC and the proposed scheme

	3DMC		proposed scheme	
	texture size	JPEG coding	texture size	JPEG coding
Coin	512x512	79 KB	512x612	91 KB
Nefertiti	512x512	96 KB	512x680	112 KB

Table 1 and Table 2 show the comparison results between the MPEG-4 3DMC standard and the proposed scheme. As we can see in Table 1, the proposed scheme had better performance than 3DMC in the texture coordinate coding, since the variances for residuals of texture coordinates along a coding order reduced after rearrangement of texture images. As a result, we could increase

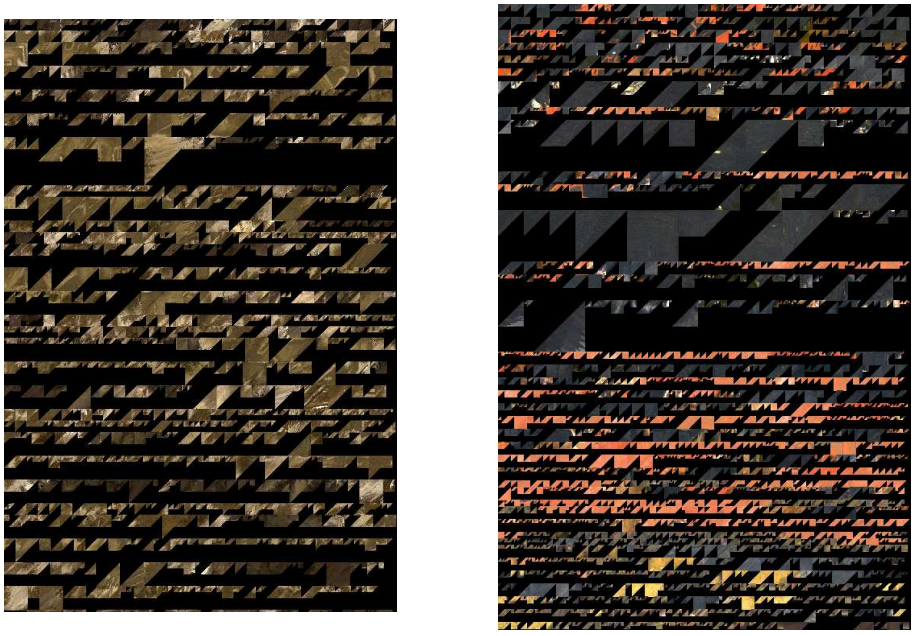


Fig. 11. Rearranged texture images

coding efficiency for texture coordinate compression by a predictive coder, and we could solve the discontinuity problem of texture coordinates for the 3-D mesh models with a non-atlas texture image.

On the other hand, the size of the rearranged texture image was larger than the size of the original texture image, because the texture image was obtained by the zig-zag allocation scheme and included holes sometimes. Rearrangement of the texture image caused the coded data for the texture images to be increased, as shown in Table 2. However, the overall coding efficiency increased by about 30% on average, since the texture coordinates had more information than a texture image. Figure 11 shows the rearranged texture images for tested models. The left image is a rearranged texture image of the Coin model and the right image is for the Nefertiti model.

5 Conclusions

In this paper, we proposed a new algorithm for 3-D mesh texture coordinate coding using a texture image rearrangement. Previous 3-D mesh compression schemes focused on the geometry and connectivity data. However, we can note that photometry information contribute substantial amount in the 3-D model data. Especially, we should have an interest to the texture coordinate coding.

Previous works related to the texture coordinate coding did not consider discontinuities in the texture coordinates. In case of the 3-D mesh model with

a non-atlas texture image, discontinuities are more serious. In this paper, we regenerated the texture image according to the texture coordinate coding order so as to remove the discontinuities in a non-atlas texture image. In order to compress the texture coordinate data, we extract the texture images and texture coordinate data from the 3-D mesh models. Then, we reallocate the texture images by the zig-zag order. Finally, we employ a predictive coder so as to compress the residual data along a vertex traversal order.

The proposed prediction coding scheme outperformed the MPEG-4 3DMC since we reduced the residual errors and eliminated the discontinuous points in texture coordinates. The proposed scheme not only overcame the discontinuity problem by regenerating a texture image, but also improved coding efficiency of texture coordinate compression. The proposed scheme can be used for the various multimedia applications needed 3-D mesh model transmission such as 3-D games and 3-D broadcasting systems.

Acknowledgements. This work was supported in part by Gwangju Institute of Science and Technology (GIST), in part by the Ministry of Information and Communication (MIC) through the Realistic Broadcasting Research Center (RBRC), and in part by the Ministry of Education (MOE) through the Brain Korea 21 (BK21) project.

References

1. Desbrun, M., Meyer, M., Alliez, P.: Intrinsic Parameterizations of Surface Meshes. Proceedings of EUROGRAPHICS (2002) 209-218
2. Derose, T., Kass, M., Truong, T.: Subdivision Surfaces in Character Animation. Proceedings of SIGGRAPH (1998) 85-94
3. Hoppe, H.: Progressive Meshes. Proceedings of SIGGRAPH (1996) 99-108
4. Garland, M., Heckbert, P.S.: Surface Simplification Using Quadric Error Metrics. Proceedings of SIGGRAPH (1997) 209-216
5. Rossignac, J.: Geometric Simplification and Compression. Course Notes 25 of SIGGRAPH (1997)
6. Deering, M.: Geometry Compression. Proceedings of SIGGRAPH (1995) 13-20
7. Taubin, G., Rossignac, J.: Geometric Compression through Topological Surgery. ACM Transactions on Graphics (1998) Vol. 17 84-115
8. Touma, C., Gotsman, C.: Triangle Mesh Compression. Proceedings of Graphics Interface (1998) 26-34
9. Isenburg, M., Sneoink, J.: Compressing Texture Coordinates with Selective Linear Prediction. Proceedings of Graphics Interface (2003) 126-131
10. Hartman, J., Wernecke, J.: The VRML 2.0 Handbook. Addison-Welsey Publishing Company (1996)
11. Yan, Z., Kumar, S., Li, J., Kuo, C-C.J.: Robust Coding of 3D Graphics Models using Mesh Segmentation and Data Partitioning. Proceedings of IEEE International Conference on Image Processing (1999) 25-28
12. Kim, S.Y., Ahn J.H., Ho, Y.S.: View-dependent Transmission of Three-dimensional Mesh Models Using Hierarchical Partitioning. Proceeding on Visual Commnications and Image Processing (2003) 1928-1938
13. Taubin, G., Horn, W., Lazarus, F.: The VRML Compressed Binary Format. ISO/IEC JTC1/SC29WG11 M3062(1998)