PROGRESSIVE TRANSMISSION OF POINTTEXTURE 3-D IMAGES

In-Wook Song, Sang-Uk Lee *

Signal Processing Lab
EECS, Seoul National University.

Chang-Su Kim

Media Communication Lab
ECE, Korea University.

ABSTRACT

A progressive compression and transmission algorithm for PointTexture 3-D images is proposed in this work. The proposed algorithm represents a PointTexture image hierarchically using an octree. The geometry information in octree nodes is encoded by the predictive partial matching (PPM) method, while the color information is encoded using the discrete cosine transform (DCT). The encoder achieves the progressive transmission of the 3-D image by transmitting the octree nodes in a top-down manner. We develop a transmission scheme, based on the rate-distortion (R-D) optimization in order to maximize the image quality subject to a given bit budget. Extensive simulation results demonstrate that the proposed algorithm is an efficient method for progressive transmission of 3-D data.

1. INTRODUCTION

Three-dimensional (3-D) data are widely used in various applications, such as virtual reality, video game and animation. Among various representation tools, the mesh representation has been a dominant method to represent 3-D data. However, photo-realistic 3-D meshes require a huge amount of storage space in general, and their distribution over digital communication channels is limited by the available bandwidth. Thus, it has drawn a lot of attention to develop alternative 3-D representation methods, which are more efficient than the mesh representation [1, 2, 3].

Depth image-based representation (DIBR) is a new approach to represent and render 3-D objects with complex geometries [1]. It is related to both the volume representation [2] and the point representation [3]. Instead of representing objects with polygonal meshes, DIBR represents a 3-D object with a set of reference images covering its visible surface. Each reference image comes with a depth map, which is an array of distances from the pixels in the image plane to the object surface. Shade *et al.* [4] proposed a DIBR method, called layered depth image (LDI). An LDI has a single reference image, but its each pixel can represent multiple points along each line of sight. Chang *et al.* [5] investigated a hierarchical representation of 3-D objects based on the LDI method.

DIBR has been adopted into MPEG-4 Animation Framework eXtension (AFX) [1, 6]. It has three formats: SimpleTexture, PointTexture, and OctreeImage. Among them, PointTexture is similar to LDI and has several advantages over the traditional mesh representation. For example, PointTexture can represent photo-realistic 3D objects without complex mesh structures. Also, its rendering complexity depends only on the image resolution, regardless of the object complexity. To render a PointTexture image, each point is simply drawn as a circular disk with its own color [1]. Thus, the rendering of PointTexture images is faster than that of triangular meshes in general. However, complex PointTexture images require a large amount of data and their compression should be performed efficiently. In this work, we propose a novel algorithm to compress PointTexture images progressively. An octree structure is employed to represent PointTexture images hierarchically.

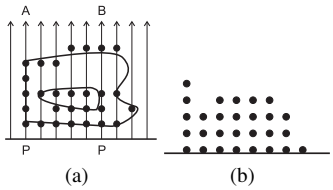


Fig. 1. An example of PointTexture: (a) virtual rays and (b) layers.

The proposed algorithm encodes the geometry information in the octree nodes using the predictive partial matching (PPM) scheme [7], which employs preceding data as a context to exploit the correlation in 3-D shapes. Also, the proposed algorithm encodes the color information based on the discrete cosine transform (DCT). The proposed algorithm supports flexible transmission of PointTexture images. Specifically, we illustrate a progressive transmission scheme, based on the rate-distortion (R-D) optimization.

This paper is organized as follows. Section 2 briefly reviews the formats and data structures of PointTexture images. Section 3 describes the octree generation algorithm. Sections 4 proposes the compression algorithm for node data, and Section 5 develops two adaptive transmission schemes for node data. Section 6 provides simulation results. Finally, conclusions are drawn in Section 7.

2. POINTTEXTURE IMAGES

There are various methods to represent and render depth images or images with depth information, for example, [4, 5, 8]. DIBR has been developed to standardize such representation methods and adopted into MPEG-4 Part 16: Animation Framework eXtension (AFX) [1, 6]. It has three main formats: SimpleTexture, PointTexture and OctreeImage. Note that SimpleTexture and PointTexture, respectively, are similar to 'sprite with depth' and LDI in [4].

PointTexture is a powerful data structure. It records the information viewed from a single camera position but allows more than one pixels along each line of sight. Each pixel has a depth value and (r, g, b) components. As shown in Fig. 1(a), each virtual ray intersects the object at multiple points, which are ordered from front to back. For example, there are five intersecting points for ray PA and four intersecting points for ray PB. The first intersecting points of all rays constitute the first layer, the second intersecting points con-

*This work was supported partly by Samsung Advanced Institute of Technology and partly by a Korea University Grant.

stitute the second layer, and so on. Fig. 1(b) illustrates those layers. From the original camera position, only the pixels in the first layer are visible. However, as a viewer moves away from the original position, the rendering system can expose back layer pixels, yielding realistic parallax.

3. HIERARCHICAL REPRESENTATION OF POINTTEXTURE IMAGES

In this section, we describe how to convert a PointTexture image into octree-based hierarchical data, which represent the 3-D image at multiple levels of details.

First, a PointTexture image is converted into volume data [2], where objects and empty regions are represented by '1' and '0' voxels, respectively. Then, the volume data are represented by an octree. Note that the octree is used as an intermediate data structure to improve the compression performance for PointTexture images. It is not related to OctreeImage, which is another format of DIBR.

3.1. Octree Generation

The volume data is expressed by an octree, whose each node is classified into one of the four categories. First, if the bounding volume contains an object, the root node is labeled with 'S' and the volume is subdivided into eight equal size volumes. If a subdivided volume contains only black voxels or only white voxels, the corresponding node is labeled with 'B' or 'W,' respectively. Otherwise, the node is set to 'S' and the volume is further subdivided into eight smaller volumes. Each color component of an S or B node is set to the average of the corresponding color components of all descendant black voxels.

This subdivision procedure can be repeated until the tree reaches the predefined maximum depth. Otherwise, the number of nodes in the tree can be too large and the compression performance of the proposed algorithm can be degraded. At the maximum depth, if a node contains both black and white voxels, it is labeled with 'P' and its voxel values are encoded by the PPM method. The PPM method will be described in detail in Section 4.2.

3.2. Progressive Transmission

We achieve the progressive transmission of a 3-D PointTexture image by transmitting the octree nodes in a top-down manner. A node can be transmitted only if its parent node has been already transmitted. Suppose that a leaf node in a partially transmitted octree is an S node. Then, it is temporally labeled as a B node, *i.e.*, all its voxel values are approximated to be black. Therefore, a node is defined to be transmittable if it is a leaf B node. At each time instance, the encoder selects one node among the set of transmittable nodes and then transmits its information.

4. COMPRESSION OF NODE DATA

The encoder uses an ordered list to maintain the set of transmittable nodes. At each time instance, the encoder selects a node from the list adaptively according to the requirements of applications, as will be described in Section 5, and then transmits its information to the decoder.

Fig. 2 shows the node data structure, which consists of header information and detailed information bits (DIB). The header information is composed of POSITION and T data. POSITION indicates the order of the current node in the list of transmittable nodes. Based on this information, the decoder can know which black node should be split or PPM-decoded. If there is only a single transmittable node, POSITION is not necessary and thus not encoded. T is a termination flag, which indicates whether the current node is the last one in the compressed bitstream or not. Note that we can avoid the use of termination flag by specifying the number of nodes at the start of the bitstream. However, this requires a preprocessing step or two-pass encoding to determine the number of nodes in the whole octree.

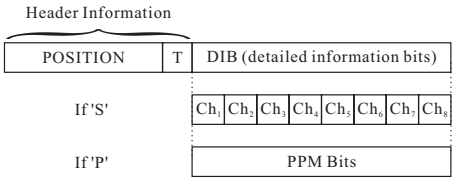


Fig. 2. Data structure for a node

As mentioned previously, the node type is determined according to its depth: a node is transmitted as a P node if it is at the maximum depth, and as an S node otherwise. Let us describe the compression methods for S and P node data subsequently.

4.1. Compression of S Node Data

For an S node, DIB consists of the geometry and color data of the child nodes. In Fig. 2, Ch_n denotes the geometry and color of the nth child. It first indicates whether the child is a B or W node. If it is B, Ch_n also contains the (r, g, b) color components, which are DPCM-encoded. Specifically, the (r, g, b) components of the child node are predicted from those of the current node, respectively, and the prediction residuals are encoded with the arithmetic coder in [9].

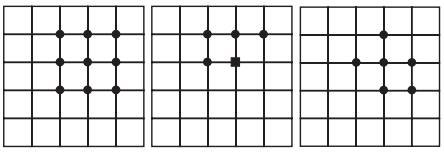
4.2. Compression of P Node Data

A P node corresponds to a cube region in the volume data. Thus, the geometry information of the P node is given by binary voxel values within the cube. In this work, we encode those binary voxel values based on the PPM method. Specifically, we encode each voxel value using the neighboring voxel values as contexts.

Fig. 3 illustrates examples of contexts. The square voxel is to be encoded, and the values of the 13 circular voxels are used as the context shown in Fig. 3(a) and (b). Since the encoding is performed in the raster scan order, those 13 voxels are causal neighbors of the current voxel. Typically, black voxels form a locally smooth surface in the 3-D space. Thus, the current voxel value can be estimated with high accuracy from the neighboring voxel values. This property is exploited by using the context-based adaptive arithmetic coder in [9].

Let n_c be the number of voxels, whose values are used as contexts. As n_c increases, the compression performance improves in general. However, the number of possible contexts is 2^{n_c}, which incurs too high complexity even for a small value of n_c. Therefore, n_c should be determined in consideration of both coding gain and implementation complexity. In this work, we set n_c to 10 as a tradeoff between coding efficiency and complexity. If $n_c = 10$, there are 286 ($= {}_{13}\mathrm{C}_{10}$) methods to retain 10 context voxels from the 13 neighboring voxels. Fig. 3(c) provides the best configuration that provide the least conditional entropies which is estimated from extensive simulations.

The color information of P nodes is encoded by a DCT-based algorithm, which is similar to the JPEG standard [10]. First, we



(a) depth = $k-1$ (b) depth = k (c) $n_c = 10$

Fig. 3. An example of contexts: The square voxel in (b) is to be encoded and the circular voxel values in (a) and (b) can be used as the context. The best configuration of context voxels in the front plane (depth = $k-1$) is shown in (c), when n_c is 10. The four additional context voxels in (b) are also used for this configuration.

form an 1-D sequence by gathering the black voxels within a P node in the raster scan order. Second, the (r, g, b) color components of a black voxel are converted into the (Y, C_b, C_r) components. Third, the 1-D sequence for each Y, C_b or C_r component is transformed by DCT. Fourth, the DCT coefficients are quantized by the JPEG quantization matrix. The quantization matrix is controlled by the JPEG quality factor, which ranges from 0 to 100. A higher quality factor corresponds to better image quality. Finally, the quantized coefficients are run-length encoded using a Huffman codeword table.

For a P node, the color information requires a much higher bit rate than the geometry information. But, the contribution of a color bit to the rendered image quality is not as big as that of a geometry bit. This is because the node is reconstructed as a visually annoying bulky cube if the geometry data are not transmitted. In contrast, the color approximation errors are less disturbing to the human visual system. Therefore, it is not effective to transmit the geometry data and the color data together as DIB. Instead, in this work, only the PPM-encoded geometry data are transmitted as DIB. After the geometry data of all P nodes are transmitted, the less important color data are transmitted.

5. TREE NODE TRANSMISSION

In this work, a PointTexture image is represented hierarchically by an octree, and the octree nodes are transmitted in a top-down manner to support progressive reconstruction at the decoder. The encoder can select the transmission order of octree nodes adaptively according to the requirements of applications. We propose a optimization scheme to determine the transmission order: rate-distortion (R-D) optimization.

5.1. Rate-Distortion Optimization

In this scheme, the transmission order is chosen to maximize the quality of the reconstructed model at a given bit budget. At each step, we select the current node from the list of transmittable nodes, which maximizes the ratio

$$-\frac{\Delta D}{\Delta R} = -\frac{D_a - D_b}{R_a - R_b}, \tag{1}$$

where $\Delta R = R_a - R_b$ denotes the difference between the rate after transmission (R_a) and the rate before transmission (R_b). Similarly, $\Delta D = D_a - D_b$ denotes the distortion difference.

To compute ΔD in (1), we calculate the geometry distortion D_g and the color distortion D_c. As a measure for the geometry distortion, we adopt the Hausdorff distance [11]. Let $\mathcal{V}$ and $\hat{\mathcal{V}}$ denote

(a) Angel (b) Flower (c) Avatar (d) Dog

Fig. 4. The all models have the resolution of $256 \times 256 \times 256$.

the sets of coordinates of black voxels in the original model and the approximated model, respectively. Then, the geometry distortion is defined as

$$D_g = h(\hat{\mathcal{V}}, \mathcal{V}) = \max_{\hat{\mathbf{v}} \in \hat{\mathcal{V}}} \min_{\mathbf{v} \in \mathcal{V}} \|\hat{\mathbf{v}} - \mathbf{v}\|, \tag{2}$$

where $\|\cdot\|$ means the Euclidean distance between two coordinates. For the color distortion, we also use the Euclidean distance between color vectors, and define

$$D_c = \sum_{\mathbf{v} \in \mathcal{V}} \|\mathbf{C}(\mathbf{v}) - \hat{\mathbf{C}}(\mathbf{v})\|, \tag{3}$$

where $\mathbf{C}$ and $\hat{\mathbf{C}}$ denote the color vectors of the original model and the approximated model, respectively. Note that $\mathcal{V}$ is always a subset of $\hat{\mathcal{V}}$ in this work, since P or S nodes are approximated by B nodes, and that the summation in (3) is carried out over $\mathcal{V}$ only.

The overall distortion D is defined as the weighted sum of the geometry distortion and the color distortion

$$D = w_g D_g + w_c D_c, \tag{4}$$

where w_g and w_c are weighting coefficients.

To compute ΔR in (1), we need a method to calculate the required bits for S and P nodes. Except the Huffman coding of DCT coefficients for P nodes, all the other data are encoded using an arithmetic coder. The exact calculation of arithmetic-coded bit rate is too complex in practice. Therefore, the entropy of each syntax item is estimated from training images and is used instead of the arithmetic-coded bit rate. For the DCT coefficients, the exact codeword lengths are easily obtained with table lookup operations.

6. SIMULATION RESULTS

The performance of the proposed algorithm is evaluated using test models in Fig. 4. The models have the resolution of $256 \times 256 \times 256$, In this work, the tree depth is selected such that the size of leaf nodes is $4 \times 4 \times 4$.

Figs. 5 illustrate the progressive transmission of the "Flower" model. In this test, the transmission order of nodes is determined by the R-D optimization scheme in Section 5.1. Note that the distortion is computed by summing up the geometry distortion and the color distortion with the weighting coefficients w_g and w_c in (4). We evaluate three combinations: $(w_g, w_c) = (1, 0)$, $(0, 1)$ and $(1, 1)$. For the "Flower" model, the proposed algorithm provides the best performance at $(w_g, w_c) = (1, 0)$. We see clear differences when 20% of the bitstreams are transmitted. This is because the "Flower" model has a detailed, complex shape. In such a case, the geometry information plays a more important role in the rendered image quality than the color information. Simulation results on various other models confirm that the performance of the proposed algorithm is

Table 1. The compression performance of the proposed algorithm, when the quality factor is set to 80. GF represents the compressed file size for geometry data only, while TF represent the file size for the whole (geometry+color) data (in bytes). PCR denotes the file size of the algorithm in [12] for geometry data. MPEG denotes the file size of the algorithm in [6] for the whole data.

	Proposed			PCR (C)	MPEG (D)	Ratio	
	GF (A)	TF (B)	PSNR			A/C	B/D
Angel	38,624	137,971	36.88	117,194	247,111	0.33	0.56
Flower	17,899	59,256	34.87	62,084	75,199	0.29	0.79
Avatar	6,779	24,027	36.23	20,607	39,584	033	0.61
Dog	12,140	40,357	38.15	21,516	69,146	0.56	0.58

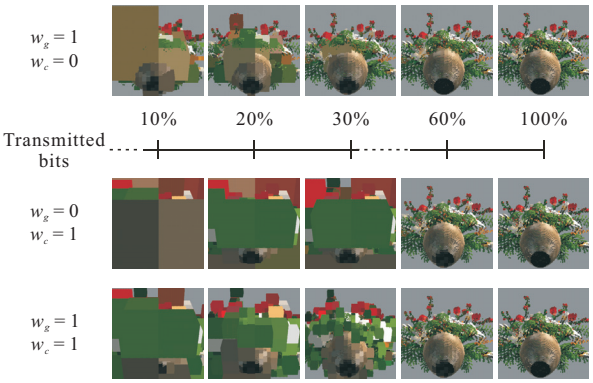


Fig. 5. Progressive transmission of the "Flower" model.

not sensitive to the weighting coefficients, as long as w_g is set to be larger than w_c.

Table 1 summarizes the compression performance of the proposed algorithm. When the whole bitstream is received, the decoder can reconstruct geometry data losslessly. However, we encode color data in a lossy manner. The quality factor is set to 80 in this test. At this high quality factor, the color reconstruction is so faithful that the reconstructed models are almost indistinguishable from the original models. For comparison, the performances of the MPEG-4 AFX algorithm [6] and the PCR algorithm [12] are also presented in Table 1. Notice that the proposed algorithm consumes only $56 \sim 79\%$ of the bit rates of the MPEG-4 AFX scheme, by allowing negligible losses in the color data. The PCR algorithm is a geometry compression scheme for voxel surfaces, which provides high coding gains if input surfaces are thin and smooth. It is observed that the proposed algorithm consumes only $29 \sim 56$ % of the bit rates of PCR for geometry coding. The coding gain is especially high for complex shapes, such as the "Flower" model. Moreover, the proposed algorithm supports progressive transmission, while PCR cannot.

7. CONCLUSIONS

In this paper, we proposed a progressive compression and transmission algorithm for PointTexture images, whose each pixel is associated with a depth value as well as a color vector. The proposed algorithm represents a PointTexture hierarchically using an octree. The octree nodes are transmitted in a top-down manner so that the decoder can reconstruct the 3-D data from coarse to fine resolutions. We developed the R-D optimized scheme for determining the transmission order of octree nodes, which can maximize the rendered image quality at a given bit budget. It was shown by extensive simulations that the proposed algorithm efficiently supports progressive transmission of PointTexture images.

8. REFERENCES

[1] Leonid Levkovich-Maslyuk, Alexey Ignatenko, Alexander Zhirkov, Anton Konushin, In Kyu Park, Mahnjin Han, and Yuri Bayakovski, "Depth image-based representation and compression for static and animated 3D objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 7, pp. 1032–1045, July 2004.

[2] A. Kaufman, D. Cohen, and R. Yagel, "Volume graphics," *Proc. IEEE Computer*, vol. 26, pp. 51–64, July 1993.

[3] M. Levoy and T. Whitted, "The use of points as a display primitive," Technical Report TR 85-022, University of North Carolina at Chapel Hill, 1985.

[4] J. Shade, S. Gortler, L. He, and R. Szeliski, "Layered depth images," in *Proc. SIGGRAPH*, July 1998, pp. 231–242.

[5] C. Chang, G. Bishop, and A. Lastra, "LDI tree: A hierarchical representation for image-based rendering," in *Proc. SIGGRAPH*, Aug. 1999, pp. 291–298.

[6] *Information Technology. Coding of Audio-Visual Objects. Part 16: Animation Framework eXtension (AFX)*, ISO/IEC Std. JTC1/SC29/WG11 14 496.16, 2003.

[7] K. Sayood, *Introduction to Data Compression*, Academic Press, second edition, 2000.

[8] M. M. Oliveira, G. Bishop, and D. McAllister, "Relief textures mapping," in *Proc. SIGGRAPH*, July 2000, pp. 359–368.

[9] P. G. Howard and J. S. Vitter, "Arithmetic coding for data compression," *Proc. IEEE*, vol. 82, no. 6, pp. 857–865, June 1994.

[10] *Information Technology – Digital Compression and Coding of Continuous-Tone Still Images*, ISO/IEC Std. JTC 1/SC 29/WG 1 N993 Recommendation T.84 ISO/IEC cd 10918-3, Nov. 1994.

[11] D. Huttenlocher, D. Klanderman, and A. Rucklige, "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, September 1993.

[12] Chang-Su Kim and Sang-Uk Lee, "Compact encoding of 3D voxel surfaces based on pattern code representation," *IEEE Trans. Image Proc.*, vol. 11, no. 8, pp. 932–943, Aug. 2002.