A 3D TEXTURE SYNTHESIS APPROACH

Ya-Lin Su, Chin-Chen Chang, Zen-Chung Shih***

Institute of Multimedia Engineering, National Chiao Tung University
Hsinchu 300, Taiwan (ROC)
*Department of Computer Science and Information Engineering, National United University
Miaoli 360, Taiwan (ROC)
E-mail: ccchang@nuu.edu.tw
**Department of Computer Science and Information Engineering, National Chiao Tung University
Hsinchu 300, Taiwan (ROC)

ABSTRACT

In this paper, a new approach for solid texture synthesis from input volume data is presented. In the pre-process, feature vectors and a similarity set were constructed for input volume data. The feature vectors were used to construct neighboring vectors for more accurate neighborhood matching. The similarity set which recorded 3 candidates for each voxel helped more effective neighborhood matching. In the synthesis process, the pyramid synthesis method was used to synthesize solid textures from coarse to fine level. The results of the proposed approach were satisfactory.

Keywords: texture, texture synthesis, solid texture, texture control.

1. INTRODUCTION

There are many different techniques for 3D surface texturing, such as texture mapping [7], [20], [21], procedural texturing [4], [14] and image-based surface texturing [17], [18], [19]. Texture mapping is the easiest way for 3D surface texturing. However, it has the common problems of distortion, discontinuity, and unwanted seams. Procedural texturing can generate high quality 3D surface textures without distortion and discontinuity, but there are still some problems. First, procedural texturing models exclude some textures, such as marble. Second, there are too many parameters for users to know and control. Results depend on the designers. Image-based surface texturing synthesizes a wider range of textures, but it fails with large structural textures such as bricks. Further, it also distorts when the curvature is too large. As a result, when 2D textures are used in texturing 3D objects, there are some disadvantages such as discontinuity, distortion on large-curvature surfaces, and non-reusability. Thus, textures generated for one surface cannot be used for other surfaces.

Solid textures can be used to overcome these problems. Peachey [13] and Perlin [14] introduced the idea of solid textures being blocks of colored points in 3D space to represent real-world materials. Solid textures obviate the need for finding a parameterization for the surface of the object to be textured while avoiding the problems of distortion and discontinuity. Moreover, solid textures provide texture information not only on surfaces, but also inside the entire volume. Several methods [3], [10], [15] use three orthogonal slices for neighborhood matching, but there are some drawbacks in these methods: They do not include the neighborhood information in 3D space and they are difficult to control in 3D space.

In this paper, an approach for real 3D space texture synthesis based on input volume data is presented. Information-rich appearance vectors and cube neighborhoods are used for neighborhood matching. The results show that the proposed approach can model a wide range of solid textures.

The rest of this paper is organized as follows: In Section 2, works related to solid texture synthesis are reviewed. In Section 3, the proposed approach for synthesizing solid textures from input volume textures is presented. Results are given in Section 4. Finally, conclusions are discussed in Section 5.

2. RELATED WORKS

Ashikhmin [1] presented an algorithm using an interactive painting-style interface for control over the texture synthesis process. Lefebvre and Hoppe [11] introduced a high-quality pyramid synthesis algorithm to achieve parallelism. Their method includes an upsampling step to maintain patch coherence, jittering of exemplar coordinates to make the texture varied, and an order-independent correction approach to improve texture quality. Lefebvre and Hoppe [12] presented a framework for exemplar-based texture synthesis with anisometric control. They used appearance vectors to replace traditional RGB color values for neighborhood matching. Their appearance space makes the synthesis more efficient because it reduces runtime neighborhood vectors from 5×5 grids to only 4 locations. They also combined their pyramid synthesis with this method to accelerate neighborhood matching and introduced novel techniques for coherent anisometric synthesis which reproduces arbitrary affine deformations in textures. They provided a convenient method for texture control. Kwatra et al. [8] presented a method for flow control on 2D textures. They presented an algorithm to achieve texture control on 3D surfaces [9].

Jagnow et al. [6] presented a stereological technique for solid textures. This approach uses traditional stereological methods to synthesize 3D solid textures from 2D images. They synthesized solid textures for spherical particles and then extended the technique by applying it to particles of arbitrary shapes. Their approach needs cross-section images to record the distribution of circle sizes on 2D slices and build the relationship of 2D profile density and 3D particle density. Users can use the particle density to reconstruct the volume data by adding one particle at a time, meaning this step is manual. This method uses many 2D profiles to construct 3D density for volume results. Their results are good for marble textures, but their system is not automatic and only for particle textures. Chiou and Yang [2] improved this method to an automatic process, but still only for particle textures.

Qin et al. [15] presented an image-based solid texturing based on a basic gray-level aura matrices (BGLAMs) framework. They used BGLAMs rather than traditional gray-level histograms for neighborhood matching. They created aura matrices from input exemplars and then generated a solid texture from multiple view directions. For each voxel in the volume, they only considered the pixels on the three orthogonal slices for neighborhood matching. Their system is fully automatic and requires no user interaction in the process. Furthermore, they can generate faithful results for both stochastic and structural textures. However, they needed large storages for large matrices and their results are not good for color textures.

Kopf et al. [10] introduced a solid texture synthesis method from 2D exemplars. They extended 2D texture optimization techniques to synthesize 3D solid textures and then used the optimization approach with histogram matching to preserve global statistical properties. They only considered the neighborhood coherence in three orthogonal slices for one voxel and iteratively increased the similarity between the solid textures and the exemplar. Their approach generates good results for a wide range of textures.

Takayama et. al. [16] presented a method for filling a model with anisotropic textures. They specified volume textures to map to 3D objects. They pasted solid texture exemplars repeatedly on the 3D object. Users can design volumetric tensor fields over the mesh and the texture patches are placed according to these fields.

3. SOLID SYNTHESIS PROCESS

3.1 Feature Vector Generation

In the proposed approach, volume data values in color space are transformed into feature vectors in appearance space. The information-rich feature vectors for each voxel are used to synthesize high-quality and efficient solid textures.

As shown in Fig. 1, after getting the RGB color values of an input volume data, the values in 5×5×5 grids for each voxel in an input volume exemplar V are used to construct feature vectors to form an appearance-space exemplar V'. There are 375 dimensions (125 for grids and 3 for RGB) for each voxel in V'. Then, PCA is performed to map V' onto a low-dimensional transformed exemplar $\tilde{V}'$.

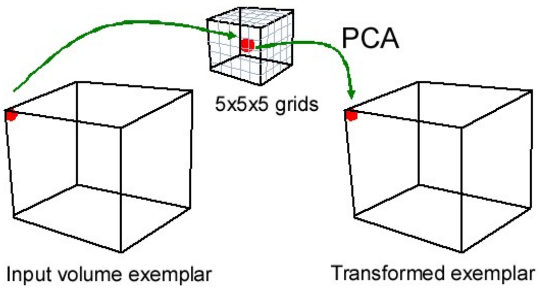

Fig. 1. Feature vector generation from an input volume exemplar V to a transformed exemplar $\tilde{V}'$.

3.2 Similarity Set Generation

Based on the k-coherence search method [21], the k most similar voxels from all voxels in the transformed exemplar $\tilde{V}'$ for each voxel p can be obtained. Then, the candidate set $C^l_{1...k}(p)$ to record the k candidates similar to voxel p is constructed, where l is a pyramid level, $C^l_1(p) = p$ and k is a user-defined parameter.

Based on the principle of coherence synthesis [1], the k most similar voxels from the $n \times n \times n$ neighborhoods of voxel p in the transformed exemplar $\tilde{V}'$ can be obtained to construct the similarity set $C^l_{1...k}(p)$ for voxel p, where n is a user-defined parameter to control the window size for coherent synthesis. However, it has the local minimum problem because global optimization is not considered. To avoid the local minimum problem, after finding $C^l_1(p)$, there is a restriction which means that the voxel in the $n \times n \times n$ neighborhoods of $C^l_1(p)$ can not be $C^l_2(p)$ and the other voxels searched are $C^l_2(p)$. By the same way, the voxel in the $n \times n \times n$ neighborhoods of $C^l_n(p)$ cannot be $C^l_{n+1}(p)$ for voxel p until $C^l_{1...k}(p)$ is constructed.

3.3 Pyramid Solid Texture Synthesis

3.3.1 Pyramid Upsampling

Based on the pyramid synthesis method [5], the proposed approach synthesizes from one voxel to a m×m×m solid texture, $S_0 \sim S_L$, where $L = \log_2 m$. A volume data S is synthesized in which each voxel $S[p]$ stores the coordinates of an exemplar voxel. First, a voxel is built and assigned the value (1,1,1) as the coordinate. Then, the coordinates of parent voxels for the next level are upsampled. Each of the eight children is assigned the parent coordinates plus a child-dependent offset as follows:

$$S_l[2p + \Delta] = S_{l-1}[p] + h_l \Delta \,,$$

$$\Delta \in \{ \begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}, \begin{pmatrix}1\\1\\1\end{pmatrix} \},$$

where h_l denotes the regular output spacing of exemplar coordinates in level l and $h_l = 2^{(\log_2 m - l)}$; Δ denotes the relative locations for 8 children.

3.3.2 Jitter Method

After upsampling the coordinates, the upsampled coordinates are jittered to achieve deterministic randomness. The upsampled coordinates at each level are perturbed by

$$S_l[p] = S_l[p] + J_l(p),$$

where $J_l(p) = h_l H(p) r_l$ is a jitter function produced by a hash function $H(p): Z^2 \to [-1,+1]^2$ and a user-defined per-level parameter r_l.

3.3.3 Voxel Correction

To make the jittered coordinates similar to those in exemplar V, the jittered coordinates are used to recreate neighborhoods. There is a feature value for each voxel after constructing feature vectors. For each voxel p, the feature values of its neighborhoods at the current level are gathered to obtain a neighborhood vector $N_{S_l}(p)$. Then, the most similar voxel is searched from the transformed exemplar $\tilde{V}'$ to make the result similar to exemplar V. In neighborhood matching, 8 diagonal locations for voxel p are used to obtain the neighborhood vector $N_{S_l}(p)$:

$$N_{S_l}(p) = \left\{ \tilde{V}'[S[p+\Delta]] \middle| \Delta = \begin{pmatrix}\pm 1\\ \pm 1\\ \pm 1\end{pmatrix} \right\}.$$

Then, the proposed approach applies Lefebvre and Hoppe's approach [12] to perform 3D coordinate correction. First, the feature values from 4 synthesized voxels near neighboring voxels of voxel p, $p+\Delta$, are averaged as the new feature value of voxel $p+\Delta$. The averaged feature value $N_{S_l}(p;\Delta)$ of voxel $p+\Delta$ is computed by

$$N_{S_l}(p;\Delta) = \frac{1}{4} \sum_{\Delta' = M\Delta, M \in \Psi} \tilde{V}'[S[p+\Delta+\Delta'] - \Delta'],$$

where

$$\Psi \in \{ \begin{pmatrix}0&0&0\\0&0&0\\0&0&0\end{pmatrix}, \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}, \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}, \begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix} \}.$$

Second, the new feature values from 8 diagonal voxels are used to construct neighborhood vectors $N_{S_l}(p)$.

A voxel u which is most similar to voxel p is searched by comparing neighborhood vectors $N_{S_l}(p)$ and $N_{S_l}(u)$. The similarity sets and coherence synthesis method are used in the searching process. The 8 neighboring voxels of voxel p are used to infer where voxel u is.

In the same way, there are 8 neighboring voxels of voxel p and each of them has 3 similar voxels. Therefore, 24 candidates are inferred for voxel p. Then, the 24 $N_{S_l}(u)$s are computed, where u is a candidate. These $N_{S_l}(u)$s are compared with $N_{S_l}(p)$ for neighborhood matching and the most similar voxel u is found to replace voxel p.

4. RESULTS

To evaluate the effectiveness of the proposed approach, several experiments were done. The proposed algorithm was implemented in MATLAB with a PC with 2.67GHz and 2.66GHz Core2 Quad CPU and 4.0GB of system memory.

Fig. 2 shows (a) an input volume data (Case_1), (b) the cross section at X=32, Y=32, and Z=32 for the input volume data, (c) a resulting volume data and (d) the cross section at X=64, Y=64, and Z=64 for the resulting volume data. The input volume data was a stochastic and marble-like solid texture. It only contained two vivid colors. It was information-rich, only needing a small amount of data to represent the whole texture. The result was continuous and not the duplication of the input data.

Fig. 3 shows (a) an input volume data (Case_2), (b) the cross section at X=32, Y=32, and Z=32 for the input volume data, (c) a resulting volume data and (d) the cross section at X=64, Y=64, and Z=64 for the resulting volume data. The input volume data was a particle-like solid texture. It contained few colors and there was a clear difference between particles and background. As long as there were few complete particle patterns in the input volume data, desirable synthesized results were obtained.

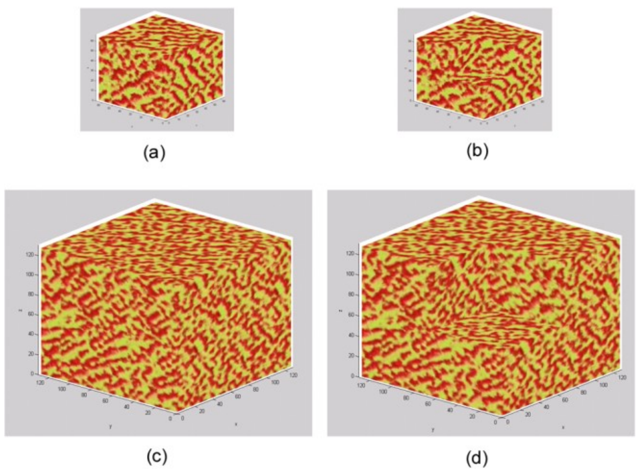

Fig. 2. (a) input volume data (Case_1), (b) cross section at X=32, Y=32, and Z=32 for input volume data, (c) resulting volume data and (d) cross section at X=64, Y=64, and

Z=64 for resulting volume data.

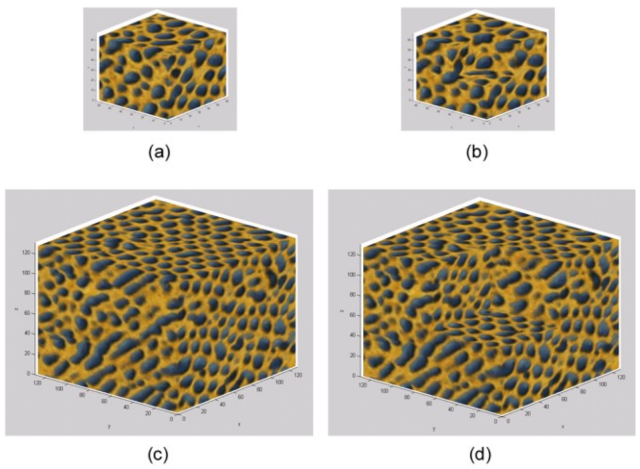
Fig. 3. (a) input volume data (Case_2), (b) cross section at X=32, Y=32, and Z=32 for input volume data, (c) resulting volume data and (d) cross section at X=64, Y=64, and Z=64 for resulting volume data.

5. CONCLUSIONS

An exemplar-based approach for solid texture synthesis has been presented. The proposed approach satisfactorily synthesized solid textures from input volume textures. In future, the proposed approach was time-consuming. This problem will be addressed in future research.

ACKNOWLEDGEMENTS

The authors would like to thank the National Science Council, Taiwan (ROC), for financially supporting this research under Contract No. NSC 96-2221-E-239-027.

REFERENCES

[1] M. Ashikhmin, "Synthesizing Natural Textures," *ACM SIGGRAPH Symposium on Interactive 3D Graphics,* pp. 217-226, 2001.

[2] J.W. Chiou and C.K.Yang, "Automatic 3D Solid Texture Synthesis from a 2D Image," *Master's Thesis*, Department of Information Management, National Taiwan University of Science and Technology, 2007.

[3] J.M. Dischler, D. Ghazanfarpour, and R. Freydier, "Anisotropic Solid Texture Synthesis Using Orthogonal 2D Views," *Eurographics 1998*, vol. 17, no. 3, pp. 87-95, 1998.

[4] D.S. Ebert, F.K. Musgrave, K.P. Peachey, K. Perlin, and S. Worley, *Texturing & Modeling: A Procedural Approach*, third ed. Academic Press, 2002.

[5] D.J. Heeger, and J.R. Bergen, "Pyramid-Based Texture Analysis Synthesis," *ACM SIGGRAPH 1995*, vol. 14, no. 3, pp. 229-238, 1995.

[6] D. Jagnow, J. Dorsey, and H. Rushmeier, "Stereological Techniques for Solid Textures," *ACM SIGGRAPH 2004*, vol. 23, no. 3, pp. 329-335, 2004.

[7] V. Kraevoy, A. Sheffer, and C. Gotsman, "Matchmaker: Constructing Constrained Texture Maps," *ACM SIGGRAPH 2003*, vol. 22, no. 3, pp. 326-333, 2003.

[8] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture Optimization for Exampled-based Synthesis," *ACM SIGGRAPH 2005*, vol. 24, no. 3, 2005.

[9] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson, and M. Lin, "Texturing Fluids," *IEEE Transactions on Visualization and Computer Graphics*, vol.13, no. 5, pp.939-952, 2007.

[10] J. Kopf, C.W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.T. Wong, "Solid Texture Synthesis from 2D Exemplars," *ACM SIGGRAPH 2007*, vol. 26, no. 3, 2007.

[11] S. Lefebvre, and H. Hoppe, "Parallel Controllable Texture Synthesis," *ACM SIGGRAPH 2005*, vol. 24, no. 3, pp. 777-786, 2005.

[12] S. Lefebvre and H. Hoppe, "Appearance-space Texture Synthesis," *ACM SIGGRAPH 2006*, vol. 25, no. 3, 2006.

[13] D.R. Peachy, "Solid Texturing of Complex Surfaces," *ACM SIGGPRACH 1985*, vol. 19, no. 3, pp. 279-286, 1985.

[14] K. Perlin, "An Image Synthesizer," *ACM SIGGPRACH 1985*, vol. 19, no. 3, pp. 287-296, 1985.

[15] X. Qin and Y.H. Yang, "Aura 3D Textures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp.379-389, 2007.

[16] K. Takayama, M. Okade, T. Ijirl, and T. Igarashi, "Lapped Solid Textures: Filling a Model with Anisotropic Textures," *ACM SIGGRAPH 2008*, vol. 27, no. 3, 2008.

[17] G. Turk, "Texture Synthesis on Surfaces," *ACM SIGGRAPH 2001*, vol. 20, no. 3, pp. 347-354, 2001.

[18] L.Y. Wei and M. Levoy, "Texture Synthesis Over Arbitrary Manifold Surfaces," *ACM SIGGRAPH 2001*, vol. 20, no. 3, pp. 355-360, 2001.

[19] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin, "Texture and Shape Synthesis on Surfaces," *Eurographics Workshop on Rendering 2001*, vol. 12, pp. 301-312, 2001.

[20] S. Zelinka and M. Garland, "Interactive Texture Synthesis on Surfaces Using Jump Maps," *Eurographics Workshop on Rendering 2003,* vol. 14, pp. 90-96, 2003.

[21] K. Zhou, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.Y. Shum, "Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces," *ACM SIGGRAPH 2002*, vol. 21, no. 3, pp. 665-672, 2002.