Fast Synthesis and Rendering of BTF on Arbitrary Surfaces

Zhan Zhang[1] Yue Qi[1] Yong Hu[1]

[1]State Key Lab. of Virtual Reality Technology, Beihang University, Beijing 100083, China
E-mail:zhangzhan@vrlab.buaa.edu.cn

Abstract

This paper proposes an approach for BTF (Bidirectional Texture Function) synthesis and rendering. A BTF can be regarded as a kind of multi-dimensional texture, which preciously shows self-shadow, self-occlusion and inter-reflection appearance of object surfaces under varying view- and light-directions. However, it is difficult to use BTF on object surfaces because of its huge data and small size. Our method uses PCA algorithm to compress BTF data, and then we propose a Wang tiles based synthesis algorithm to solve its size problem. To keep consistency of multi-dimensional, we also propose an error determination algorithm for Wang tiles' construction in synthesis process. Using the method we proposed can render BTF on arbitrary object surfaces efficiently and quickly.

Keywords: Bidirectional Texture Function (BTF), PCA, Compression, Wang tiles, Synthesis, GPU

1. Introduction

Accurate modeling of real world has always gained great attention in Computer Graphics and Computer Vision. Today, geometric modeling like using image-based modeling method and 3D scanner can explicitly get geometry of object to a certain scale. In contrast, attribute modeling of surface reflection is difficult, because it needs high-dimensional function to represent light transporting process when crosses an object surface, and this brings problems to data acquisition, compression and rendering.

Bidirectional Texture Function (BTF) can solve the problem. Dana et al. [1] introduced BTF, a 6-dimensional texture representation in 1999. A BTF of certain material is represented by a set of images with different view- and light-directions. It can preciously show the self-shadow, self-occlusion and inter-reflection appearance of object surfaces when view and light changing.

Normally it needs thousands of images to represent a BTF, which makes it necessary to compress them before rendering. Also, BTF should be synthesized when used on object surfaces because BTF sample is a small plane for the limitation of acquisition equipment.

In this paper, we introduce some research works on BTF in section 2. And in section 3, we use PCA algorithm to compress BTF. In section 4, we propose a BTF synthesis method based on Wang tiles and also propose an error determination algorithm in synthesis process to keep the continuity of BTF. Finally in section 5, we give the compression, synthesis and rendering results.

2. Related Work

2.1. Bidirectional Texture Function

Using densely sampled BTF instead of ordinary 2D texture makes more realistic appearance of material. BTF research work includes these steps: acquisition, compression, and synthesis.

2.1.1. Acquisition

Dana et al. [1] was the first to use gonioreflectometer-like setup with CCD-Chips to capture spatial variance of reflection, and they created BTF database called *CUReT* [2]. Furukawa et al. [4] proposed a method using range cameras and a reconfigurable camera array to capture BTF. At the same time, Bonn University [3] designed an automatic acquisition system with an HMI (Hydrargyrum Medium Arc Length Iodide) bulb, a robot holding the sample and a rail-mounted CCD camera, which can capture BTF quickly.

2.1.2. Compression

It is not doable to render BTF using raw data because it is really huge. Van Ginnneken et al. [4] proposed a method using texture histogram to correlate texture with viewing and irradiance changes. Leung and Malik [15] proposed the concept of Texton, and used it to cluster BTF under varying view- and light-directions. Suykens [6] represented BTF as spatially variant BRDF and compressed it using chained matrix factorization (CMF). Wong et al. [7] applied Spherical Harmonic transform to compress BTF in a frequency domain. PCA can also be used in BTF compression.

2.1.3. Synthesis

For the limitation of BTF sample size, it needs to be synthesized when applied to object surfaces. Liu [8] first synthesized BTF using 2D texture synthesis methods. His method needed to compute similarity from shading shape. Tong et al. [9] improved the method and used *k-coherent* and Texton to synthesize BTF on arbitrary surfaces. Recently, Zhou et al. [10] designed an interactive system to synthesize BTF using Graph-cut methods.

2.2. Wang Tiles

The theory of Wang tiles was given in early 1960s. Hao Wang proposed that a set of tiles with colored edges could generate a plane of any size non-periodically with edges sharing same color matched. And this theory was extended to Computer Graphics these years.
Stam et al. [11] applied Wang tiles to texture creation. Cohen et al. [12] extended Wang tiles to 2D texture synthesis and proposed an automatic synthesis method. Chenny et al. [13] later applied Wang tiles to create animated flow pattern. Wang tiles were extended to contain flow information. Wei et al. [14] designed an arrangement scheme to correct the texture filtering problem across tile images. Later, Fu and Leung [16] gave a method of using Wang tiles on any topologic surfaces. Kopf [17] proposed a synthesis method using recursion algorithm. Recently, Lagae and Dutre [18] proposed a new Wang tile based method matching the corner color instead of edge color.

3. Compression

Normally, light-dependent variations of appearances with fixed views are relatively smooth and can be easily interpolated and approximated. We use PCA for this purpose, approximating light-dependent variation of object appearances and keeping view-dependent variation discrete. A certain material shows similar features under different view- and light-directions. So PCA can use a few components to reconstruct the raw BTF.

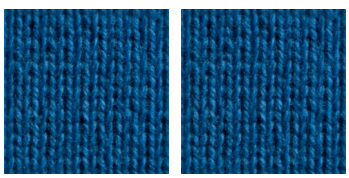
The image set of view direction j constructs the matrix T_j and its average matrix $\overline{T_j}$. We get the adjusted matrix T_j' from T_j by subtracting $\overline{T_j}$. Eigenvectors $E_1 ... E_n$ of T_j' is known as eigentextures. Using first c eigenvectors can reconstruct images under any light directions, as in equation (1).

$$N_{ij} = \sum_{k=1}^{c} \beta_{ik} E_k + \overline{T_{ij}}, \quad i = 1...n \quad (1)$$

Where N_{ij} denotes the texture in light direction i and view direction j, $\overline{T_{ij}}$ denotes the average texture in light direction i and view direction j, β_{ik} is computed as equation (2).

$$\beta_{ik} = < T_{ij}', E_k > \quad i = 1...n \quad (2)$$

Fig.1 shows the result of reconstructed image when c value chosen 5.



(a) (b)

Fig.1: reconstruction result. (a) is raw image and (b) is reconstructed image

The error between raw and reconstructed images is less than 2.22%, the compression ration is 6.17%

4. Synthesis

4.1. Overview

Certain pixels in different images have varying values which introduces inconsistent mesostructures for BTF synthesis. We represent compressed BTF as a multi-dimensional texture when synthesis to keep the consistency of each dimension. The overview of our method is depicted in Fig.2.

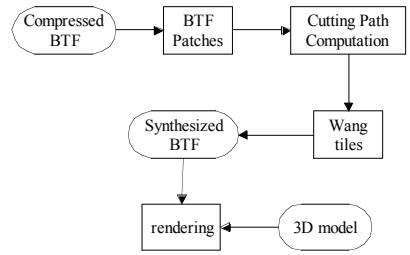


Fig.2: overview of our synthesis method

From compressed BTF random choose four patches for a tile with error $d < d_{\max}$ in the overlap region. Then compute a cutting path between every two adjacent patches. A Wang tile is generated along the cutting path. BTF can be synthesized to any size using a set of Wang tiles and rendered on object surfaces.

4.2. Error Computation

It is necessary to compute error in overlap region between two patches to keep the continuity of patches. We propose a method to determine the error threshold according to PCA, as in equation (3), (4).

$$d = \sqrt{\frac{1}{N} \sum_{v=1}^{V} \alpha_v \sum_{m=1}^{c} \beta_m \sum_{k=1}^{N} (p_k^1 - p_k^2)^2} \quad (3)$$

$$d_{\max} = \sqrt{\frac{1}{N} \sum_{v=1}^{V} \alpha_v \sum_{m=1}^{c} \beta_m \sum_{k=1}^{N} (\varepsilon p_k^1)^2} \quad (4)$$

N denotes the number of pixels in the overlap region, c denotes the principle components number chosen by PCA, β_m, α_v are coefficients separately in light-direction m and view-direction v, p_k^1 and p_k^2 are color of pixel k in two adjacent patches, ε is the error sensitive parameter by which to adjust the error tolerance.

Regarding BTF as a c-channel, values for each pixel represent light and view variations of the BTF. All light and view

variations are integrated into the error function by just summing up the errors for all c channels.

4.3. Cutting-path and Wang tiles Construction

We propose a Wang tile construction method for BTF and using them for fast synthesis. The construction of Wang tiles is depicted in Fig.3.

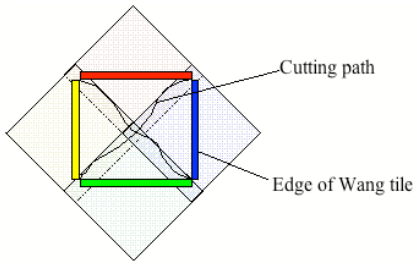


Fig.3: Construction of Wang tile

Four diamond (got by rotating 45°) patches are random chosen within error threshold. Compute cutting paths which show similarity of two patches in every overlap region and a Wang tile is constructed along cutting path's end point. In the overlap region, we use Dijkstra algorithm to determine the cutting path.

To create a continuous BTF from a small size sample, Wang tiles must be found for each fit together across the boundaries with matching colors.

4.4. Rendering

We render BTF in glsl Shader. Input is Synthesized BTF which is represented as c 2D textures. Weights got by PCA are also stored in a 2D texture.

Previously compute a delaunay triangulation of the hemispheres which distributes the acquisition directions for view and light separately.

In Vertex Shader, light and view directions for model vertexes are computed. Be sure they are in vertex coordinate system. In Fragment Shader, search the triangles view- and light- directions belongs to. Final color can be approximated by tri-linear interpolation from three vertexes, see equation (5) (6).

$$T = \sum_{i=1,j=1}^{3} vw_i lw_j T_{ij} \quad (5)$$

$$vw_1 = \frac{\begin{vmatrix} \mathrm{v} & x2 & x3 \\ v & y2 & y3 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ 1 & 1 & 1 \end{vmatrix}} \quad (6)$$

vw_i and vl_i define the weights of three vertexes. (xi, yi) is vertex coordinates. Other weights can be got similar as vw_1 . T_{ij} are the sample texture in view-direction i and light-directon j. BTF can rendered on arbitrary surfaces smoothly using this interpolation method, detailed results are included in section 5.

5. Results

We implemented our synthesis and rendering method on a PC with 3.4GHz CPU, 1G memory and 8800GT graphics board. BTF samples are from Bonn BTF database [3]: WOOL and IMPALLA. These samples are captured with 81 view- and 81 light- directions, resulting in a set of total 6,561 sample images. The models we choose are bunny and shark.

The experiment results are shown in table 1. From it can conclude that the size of texture affects little in rendering speed but much in pre-computation time. The rendering results are shown in Fig.4, (a), (d) are BTF sample IMPALLA and WOOL. (b), (c) are the results of sample IMPALLA applied to bunny and (e), (f)

are the results of sample WOOL applied to shark. We can see realistic appearance on object surfaces when light and view direction changes.

Table 1. Experiment result

Sample(model)	Sample size	Model size (triangles)	Synthesized BTF size	Pre-computation time	Fps
IMPALLA (bunny)	128*128	30660	1024*1024	44mins	106
WOOL(shark)	256*256	20588	1024*1024	32mins	134

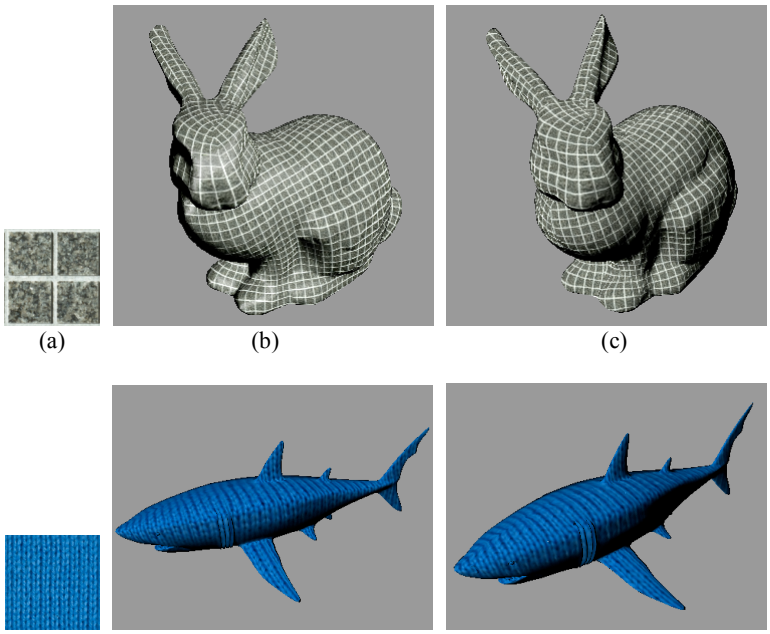


Fig.4: rendering results. (a), (d) are sample WOOL and IMPALLA. (b), (c) show the result of sample WOOL applied to bunny and (e), (f) show the result of sample IMPALLA applied to bird.

6. Conclusion

The huge data and small size makes it difficult to really use BTF. In this paper, we applied PCA to compression and proposed a BTF synthesis method, which can be used to effectively render BTF on arbitrary surfaces.

Our synthesis method is applicable to isotropic material but it won't work well to anisotropic pattern, because our method only considers edges matching.

Also, multi- material BTF may be an interesting research point in the future.

7. Acknowledgement

This paper is supported by National Nature Science Foundation of China (No. 60533070 and 60773153), the Key grant Project of Chinese Ministry of Education (No. 308004), the Project of Chinese Ministry of Science and Technology (No. 2006BAK12B09), the Project of Beijing Municipal Science and Technology

Commission (No. Z07000100560714), National High Technology Project (863 Project) (2006AA01Z333).

8. References

[1] K.J. Dana, B. van Ginneken, S.K. Nayar, and J.J. Koenderink, "Reflectance and Texture of Real World Surfaces," *ACM Trans. Graphics*, vol. 18, no. 1, pp. 1-34, Jan. 1999.

[2] http://www.cs.columbia.edu/CAVE/software/curet/in-dex.php.

[3] http://btf.cs.uni- bonn.de.

[4] R. Furukawa, H. Kawasaki, K. Ikeuchi and M. Sakauchi, "Appearance Based Object Modeling Using Texture Database: Acquisition, Compression and Rendering," *Proc. 13th Eurographics Workshop Rendering* (*EGRW 02*), pp. 257-266, 2002.

[5] B. Van Ginneken, J.J. Koenderink, and K.J. Dana, "Texture Histograms as a Function of Irradiation and Viewing Direction," *Computer Vision*, vol. 31, no. 2-3, pp. 169-184, 1999.

[6] F. Suykens, K. vom Berge, A. Lagae, and P. Dutre, "Interactive Rendering with Bidirectional Texture Functions," *Computer Graphics Forum*, vol. 22, no. 3, Sept. 2003.

[7] P.M. Lam, C.S. Leung, and T.T. Wong, "Noise Resistant Fitting for Spherical Harmonics," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 2, pp. 254-265, Mar./Apr. 2006.

[8] X. Liu, Y. Yu, and H. Y. Shum, "Synthesizing Bidirectional Texture Functions for Real-World Surfaces," *Proc. Int. Conf. Computer Graphics* (*SIG- GRAPH 01*), pp. 97-106, 2001.

[9] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, and H. Y. Shum, "Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 665-672, 2002.

[10] K. Zhou, P. Du, L. Wang, J. Shi, B. Guo, and H. Y. Shum, "Decorating Surfaces with Bidirectional Texture Functions," *IEEE Trans. Visualization and Computer Graphics*, vol. 11, no. 5, pp. 519- 528, 2005.

[11] J. Stam, "Aperiodic Texture Mapping," Technical Report R046, *European Research Consortium for Informatics and Math.*, 1997.

[12] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen, "Wang Tiles for Image and Texture Generation," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 287-294, 2003.

[13] S. Chenney, "Flow Tiles," *Proc. ACM SIGGRAPH/ Euro-graphics Symp. Computer Animation*, pp. 233-242, 2004.

[14] L. Y. Wei, "Tile-Based Texture Mapping on Graphics Hardware," *Proc. SIGGRAPH/Eurographics Conf. Graphics Hardware*, pp. 55-63, 2004.

[15] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearance of Materials Using Three-Dimensional Textons," *Int'. J. Computer Vision*, vol. 43, no. 1, pp. 29-44, 2001.

[16] C. W. Fu and M. K. Leung, "Texture Tiling on Arbitrary Topological Surfaces Using Wang Tiles," *Proc. Eurographics Symp.Rendering* (*EGSR '05*), pp. 99-104, June 2005.

[17] J. Kopf, D. Cohen, O. Deussen, and D. Lischinski, "Recursive Wang Tiles for Real-Time Blue Noise," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 509-518, 2006.

[18] A. Lagae and P. Dutre, "An Alternative for Wang Tiles: Colored Edges versus Colored Corners," *ACM Trans. Graphics*, vol. 25,no. 4, pp. 1442-1459, Oct. 2006.