Tileable BTF

Man-Kang Leung, Wai-Man Pang, *Student, IEEE*, Chi-Wing Fu, *Member, IEEE*,
Tien-Tsin Wong, *Member, IEEE Computer Society*, and Pheng-Ann Heng, *Senior Member, IEEE*

Abstract—This paper presents a modular framework to efficiently apply the bidirectional texture functions (BTF) onto object surfaces. The basic building blocks are the BTF tiles. By constructing one set of BTF tiles, a wide variety of objects can be textured seamlessly *without* resynthesizing the BTF. The proposed framework nicely decouples the surface appearance from the geometry. With this *appearance-geometry decoupling*, one can build a library of BTF tile sets to instantaneously dress and render various objects under variable lighting and viewing conditions. The core of our framework is a novel method for synthesizing seamless high-dimensional BTF tiles, which are difficult for existing synthesis techniques. Its key is to shorten the cutting paths and broaden the choices of samples so as to increase the chance of synthesizing seamless BTF tiles. To tackle the enormous data, the tile synthesis process is performed in a *compressed* domain. This not only allows the handling of large BTF data during the synthesis, but also facilitates the compact storage of the BTF in a GPU memory during the rendering.

Index Terms—Three-dimensional graphics and realism, color, shading, shadowing, texture, picture/image generation, methodology and techniques.

1 INTRODUCTION

REALISTIC modeling and rendering of surface-light interaction is one of the major goals in computer graphics. Several reflectance models of different levels of detail such as the Bidirectional Reflectance Distribution Function (BRDF) [5], [43], Bidirectional Texture Function (BTF) [12], [20], [47], and Bidirectional Surface Scattering Reflectance Distribution Function (BSSRDF) [24], [50] have been proposed to address the problem. This paper introduces a modular framework to apply the bidirectional texture functions (BTF) in appearance modeling. Our goal is to *decouple the BTF synthesis from the surface geometry* so that changing the surface geometry does not require resynthesizing the BTF. To achieve this goal, we first construct the *BTF tiles* instead of directly synthesizing the BTF on the geometry surface.

Surface appearance modeling using the BTF can be roughly subdivided into the following phases:

1. *BTF acquisition* (real or synthetic data),
2. *BTF synthesis*,
3. *BTF compression*, and
4. *BTF rendering*.

Once the raw BTF data is acquired, existing approaches normally synthesize the BTF *directly* onto the target geometry to avoid visible cutting seams and to minimize the geometric distortion. However, as the synthesis process is applied directly onto the target geometry, the synthesized

BTF data is tied to the geometry surface and cannot be reused elsewhere. Furthermore, if we want to change the surface appearance with another BTF, we are forced to resynthesize the BTF data even for the same target surface. Rather than having a surface-dependent BTF data, the proposed framework introduces a tile space to decouple the surface geometry from the synthesis process. The decoupling is done by replacing the target geometry surface with an intermediate tile space and by synthesizing the BTF in this tile space. Fig. 1 outlines the proposed approach. Note that the proposed BTF synthesis framework is independent of the geometry. With this framework, we gain the following advantages.

Surface independence and reusability. Since the synthesized BTF tiles are independent of the surface geometry, we can efficiently synthesize the BTF tiles without referring to any particular surface geometry. The tile set can be repeatedly used for dressing a wide variety of surface models; we do not need to modify any tile or synthesize more tiles. Hence, one can construct a library of BTF tile sets and use the tile sets over and over again.

Instant redressing. Furthermore, by defining a canonical organization of tiles so that all BTF tile sets share the same tile arrangement, we can instantaneously redress a tiled surface simply by looking up another tile set. No retiling or extra computation is needed.

Aperiodic. Even when the total number of BTF tiles within a tile set is finite, the nonperiodic property is achieved via Wang tiling [9], [46], [55]. As the conventional Wang tiling is only applicable for planar domain, we employ the techniques we devised in our previous work [18] to generalize Wang tiling on surfaces with more general topologies.

Compactness. Since the BTF tiles are rectilinear in structure, they not only fit nicely into the memory, but also facilitate compression using standard block-based methods like S3 Texture Compression (S3TC). The entire BTF data is

• M.-K. Leung and C.-W. Fu are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: cskang@ust.hk, cwfu@cse.ust.hk.

• W.-M. Pang, T.-T. Wong, and P.A. Heng are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong.
E-mail: wmpang@ieee.org, ttwong@acm.org, pheng@cse.cuhk.edu.hk.

Manuscript received 27 July 2006; revised 29 Nov. 2006; accepted 21 Dec. 2006; published online 2 Feb. 2007.

For information on obtaining reprints of this article, please send e-mail to: tcvg@computer.org, and reference IEEECS Log Number TVCG-0112-0706.
Digital Object Identifier no. 10.1109/TVCG.2007.1034.

Fig. 1. Overview of the proposed framework.

compact enough to be stored in the current graphics processing unit (GPU) memory for time-critical rendering.

The core of our work is the BTF tile synthesis. Due to the *high dimensionality* and *enormous storage* of the BTF, it is hard and computationally very expensive to find a synthesis solution without any obvious seam. Unlike the color (red, green, blue (RGB)) texture synthesis involving only three dimensions, BTF synthesis usually involves up to thousands of dimensions. Although the BTF exemplar for synthesis is usually small in spatial resolution (probably due to acquisition difficulty and high storage requirements), finding a solution without obvious seams over thousands of dimensions is even more difficult. To solve the problem, we present a novel method for synthesizing high-dimensional BTF tiles. It consists of four major substeps: corner sampling, edge synthesis, frame construction, and interior area synthesis. To make the synthesis tractable, we also perform the synthesis in a *compressed* domain.

2 RELATED WORK

2.1 Bidirectional Texture Function

In computer graphics, the use of BTF involves the following processes:

1. acquisition,
2. compression,
3. synthesis, and
4. rendering.

BTF acquisition. Dana et al. [12] were the first in capturing and modeling the BTF data from real-world materials. They built the first BTF database called *CURET* [11]. Another database of higher resolution and denser sampled BTFs was recently collected by the University of Bonn [4], [36]. Furukawa et al. [19] devised an automatic method in capturing the BTF data from 3D models by using range cameras and a reconfigurable camera array at the same time. Han and Perlin [23] developed a kaleidoscope-based capturing system for fast capturing of the BTF data without mechanical movement.

BTF compression. Van Ginneken et al. [20] proposed using a texture histogram to correlate texture with viewing and irradiance changes. Leung and Malik [33] developed the 3D texton concept as clusters of filter output of textures under different viewing and lighting configurations. They further applied the method on the *CURET* data for texture recognition. Suykens et al. [48] represented the BTF as spatially variant BRDF and applied the chained matrix factorization (CMF) to decompose it and render it using the GPU. On the other hand, the PCA method [10], [26], [56]

and the multilinear method [34], [53] were also used to compactly represent the high-dimensional and gigantic BTF data. Spherical harmonics (SHs) [43] is another stream of research on compressing high-dimensional BTF. Inspired by the BRDF representation, Wong et al. [32], [60], [61] achieved compression in a frequency domain using SH transform.

BTF synthesis. Before rendering the BTF data on a target geometry surface, the BTF has to be synthesized with reference to the target geometry. Liu et al. [35] were the first in applying texture synthesis methods [14], [15], [25], [29], [37], [52], [59], [62] to produce seamless BTF. Approximated geometry was first recovered using shape from shading and then served as a guidance for the BTF synthesis process. Tong et al. [51] improved this method and synthesized the BTF on arbitrary surfaces using the *k*-coherent search method. Zhou et al. [63] presented an interactive painting system to efficiently synthesize the BTF on arbitrary surfaces using the graph cut method [29].

BTF rendering. The BTF captures the surface appearance, as well as its mesostructure; thus, it can greatly increase the surface realism in the rendering. Chen et al. [6] applied the BTF to render feathers with a controllable parametric L-system. Sattler et al. [41] captured the mesostructure of fabric by using the BTF formulation and applied the compressed BTF data to render cloth. Sloan et al. [45] applied biscale decomposition on radiance transfer so as to add global transport effects to the BTF rendering. Wang et al. [57] proposed a real-time rendering framework for plant leaves using the spatially variant BRDFs along with subsurface scattering analysis.

2.2 Wang Tiling

The core of our BTF synthesis is Wang tiling [55]. The theory of Wang tiles can be traced back to the early 1960s when Wang [55] studied whether Wang tiling is decidable. The tile set consists of a set of square tiles, known as Wang tiles, where edges of tiles are color coded. In order to create a valid tiling of a plane by Wang tiles, all shared edges should have matched color. Grünbaum and Shepherd [21] examined this subject in depth and presented the nonperiodic tiling of a plane using a finite Wang tile set.

Stam [46] was the first in applying nonperiodic Wang tiling to texture creation. Wang tiles were used as texture container for patterns such as water surface and caustic. Cohen et al. [9], [42], further investigated the use of Wang tiles in texture synthesis and invented an automatic method to synthesize textures on Wang tiles. Chenney [7] later applied tiling to create animated flow pattern. Wang tiles were extended to contain flow information. Wei [58]

Fig. 2. Compressing the BTF data using the double SH projection with the CLS method.

devised a Wang tile arrangement scheme in texture memory so as to correct the texture filtering problem across tile images. Fu and Leung [18] later generalized the conventional Wang tiling mechanism and made Wang tiling applicable to arbitrary topological surfaces. Although Fu and Leung [18] discuss only conventional texture tiling, this paper focuses on more general BTF texture tiling and introduces a novel synthesis mechanism for making high-dimensional BTF tiles. Recently, Lagae and Dutré [31] invented an alternative Wang tile set by using colored corners instead of colored edges, whereas Kopf et al. [28] developed a recursive mechanism in Wang tiling.

3 BTF TILE SYNTHESIS

The BTF is a six-dimensional (6D) function capturing the surface reflectance of a 2D texture under variable illumination and viewing configurations:

$$\text{The BTF data space} = V \times L \times T,$$

where V is the viewing domain $\{\theta_v, \phi_v\}$, L is the lighting domain $\{\theta_l, \phi_l\}$, and T is the texture (spatial) domain (u, v) . Azimuth ϕ_v (and ϕ_l) spans $[0, 2\pi)$, whereas altitude θ_v (and θ_l) spans $[0, \pi/2]$ over a hemisphere. Thus, an acquired BTF sample can be stored in the form of a 6D table of pixels or, equivalently, a four-dimensional array of texture images. Starting with a different point of view, Wong et al. [60] independently proposed the same 6D formulation known as the apparent BRDF of pixels (ABRDF) in the context of image-based rendering.

3.1 BTF Compression

Due to the enormous data volume of the BTF, processing of a plain BTF is computationally very expensive. To reduce the computational cost, we propose to synthesize the BTF in a *compressed domain*. First, we apply the SH transform on both the lighting (θ_l, ϕ_l) and viewing (θ_v, ϕ_v) dimensions, that is, double SH projection [43]. The spatial dimension (u, v) is left untouched as the subsequent BTF synthesis will perform spatial segmentation. Hence, the outcome of this encoding is a 2D array of SH matrices. Each element (u, v) maintains a $k_l \times k_v$ matrix of the SH coefficients, where k_l and k_v are the number of SH coefficients kept for lighting and viewing dimensions, respectively (see Fig. 2).

The double SH projection we employed is not obtained by spherical integration because the BTFs are normally acquired (sampled) over the upper hemisphere only, instead of the full sphere. Rather than using the least

Fig. 3. Synthesizing seamless tile by merging four diamond-shaped samples [9].

square fitting as in [44], we applied the *constrained least square (CLS) method* to estimate the *noise-proof* SH coefficients [32]. It can be shown that SH coefficients with large magnitude are very sensitive to noise introduced by modern quantization and compression techniques. The CLS method noise-proves the SH coefficients by suppressing their magnitudes. Although we apply SH transform to compress the BTF data in our current implementation, it may also be replaced by other sophisticated representations such as PCA or TensorTextures [53] because our framework is independent of the compression scheme being adopted. However, since the constrained SH projection we currently employed is noise resistant, we can minimize the visual artifact introduced by the quantization process that follows.

3.2 Synthesizing the BTF Tiles

This section introduces a novel method for synthesizing high-dimensional BTF on rectangular tiles that can facilitate the following Wang tiling. Cohen et al. [9] synthesized texture patterns on tiles using the diamond-shaped samples as depicted in Fig. 3. This method first extracts a set of diamond-shaped samples from the input texture. Then, it creates seamless tiles by merging four diamond-shaped samples side by side using dynamic programming [14] or the standard graph-cut technique [29]. Cutting paths are traced in the overlapping regions between diamond-shaped samples to avoid seam. One advantage of this method is its fast computation as only small number of diamond-shaped samples is required.

However, when applying it to high-dimensional BTF, we found that visible seams always pop up along the cutting paths on the synthesized tile images. The major reason is due to the high-dimensionality of BTF. Unlike the RGB texture that contains only three layers, the SH-projected BTF contains $k_l \times k_v \times 3$ coefficient layers, where k_v and k_l could be as small as 25 in order to achieve acceptable rendering quality. Hence, it is very difficult to always guarantee a seamless cutting path across all coefficient layers, especially when the cutting path is long.

Therefore, the key is to avoid long cutting paths. To address this issue, we divide the tile synthesis process into the following four major steps: corner sampling, edge synthesis, frame construction, and synthesis within frame. This idea was inspired by the formulation of the Poisson disk tiles [30] invented by Lagae and Dutré. Fig. 4 illustrates

Fig. 4. Four steps for synthesizing a BTF tile. Corner sampling $\rightarrow$ edge synthesis $\rightarrow$ frame construction $\rightarrow$ interior area synthesis.

these four steps. Our goal is to increase the chance of obtaining seamless merging. Note that the tile synthesis process actually works on a volume with the third dimension spanning the SH coefficients. For simplicity, we ignore this third dimension throughout the discussion below.

3.2.1 Corner Sampling

To generate a tile set, we first extract a set of small *corner samples* from the raw BTF data. For each color-coded edge in the tile set, we extract a pair of rectangular corner samples from the BTF data volume. Each rectangular corner sample has size $h/2 \times h$, provided that $(w - h) \times h$ is the size of an edge to be synthesized and $w \times w$ is the size of a resultant tile, see Fig. 5. Note that the choice of the edge height, h , corresponds to the feature size in the BTF pattern. Since red and green (blue and yellow) color-coded edges are horizontal (vertical) in nature, we pick left and right (top and bottom) rectangular corners for them. Furthermore, when extracting corner samples from the input BTF volume, we try to maintain the similarity between corresponding corner samples so that, by the time they are merged in Step 3, we can ensure a seamless cutting path between corner samples in the combined tile frame.

To be precise, the word “similarity” refers to the seamlessness of the cutting paths (from the graph cut algorithm) between the corresponding corner samples that will be merged in Step 3. In other words, we should find a set of corner samples in Step 1 so that the cutting paths to be applied in Step 3 are of very low matching error. Note that it is relatively easy to find such a set because the cutting paths here are much shorter as compared to the case of diamond-based method.

3.2.2 Edge Synthesis

The second step is to connect pairs of corner samples by synthesizing the $(w - h) \times h$ pixels in between them. The

Fig. 5. Sizes (in pixel unit) of corner samples, edge samples, and a BTF tile (from left to right).

mechanism is achieved by applying the graph-cut algorithm iteratively to extract patches from the BTF volume and fill up the blank area in between the corner samples. Fig. 6a shows the iterative synthesis of an edge. Also, we should avoid rotating the T and B corner samples as the SH coefficients are not rotation invariant without proper rotation computation.

Note that, in applying the graph-cut algorithm, we keep track of the similarity error (seamlessness) for all pixels inside the blank area to be filled. All similarity errors are set to infinity at the beginning. When a candidate patch from the raw BTF data is considered, we first overlap it with the existing filled area and apply the graph cut to find a cutting path between the patch and the filled area. Note that the error metric we used is a weighted sum of square difference between the SH coefficient vectors from the patch and the filled area. After that, the total errors along the cutting path are summed and compared against the total errors currently inside the fillable area to check if there is any improvement. Hence, if the candidate patch is good enough to be applied to fill the edge sample, the per-pixel similarity errors inside the filled area will be updated accordingly based on the errors previously found along the cutting path. This process is repeated until the whole area has been filled and the total (also individual) per-pixel similarity error falls below a certain user-defined threshold (as well as an individual threshold for each pixel).

3.2.3 Frame Construction

After synthesizing all edges, we can construct the tile frame, as illustrated in Step 3 in Fig. 4. At each tile corner, there is an overlapping area of size $h/2 \times h/2$. A cutting path in this region is determined by the path previously found in Step 1; note that, to measure the similarity (seamlessness) among

Fig. 6. (a) Edge synthesis and (b) interior area synthesis.

corner samples in Step 1, we have already applied the graph-cut algorithm between corresponding corner samples. Note again that, since the cutting paths we used here are much shorter than that in the diamond-based method, we can easily find seamless cutting paths in all our experiments, even for the high-dimensional BTF. Once all edges are joined, the extra area outside the dotted line is clipped away.

3.2.4 Interior Area Synthesis

Finally, the graph-cut filling algorithm (as illustrated in the edge synthesis part) is applied again to iteratively synthesize the interior area within the constructed frame. Fig. 6b shows the iterative synthesis of the interior area.

The reasons why this synthesis method can handle high-dimensional BTF are that it *relaxes the constraints* (shortens the cutting paths) and *increases the choices* of samples (not just restricted to four selected diamond-shaped samples). The pair of corner samples corresponding to the same color-coded edge need not be paired up horizontally or vertically in the input BTF data volume, so we have more choices for our cornerstones. As our synthesis method does not limit us to choose four diamond-shaped samples, we can have numerous choices of samples and, hence, it substantially increases the chance to obtain seamless cutting paths. In all our experiments, the proposed method can successfully determine seamless cutting paths.

3.3 User-Controllable Tile Synthesis

In addition to the fully automatic tile synthesis method presented above, we also allow the users to interactively edit the features in order to fine tune the BTF tiles. To achieve this goal, we developed the user-controllable tile synthesis GUI, as shown in Fig. 7. Via this GUI, users can drag-and-drop features from the data sample onto the BTF tiles and constrain the BTF tile synthesis process [63] to preserve this user-edited content.

To illustrate how it works, we demonstrate how to synthesize tiles with this GUI. Note that the user can change the lighting and viewing on the synthesized BTF tile during the editing, but they are fixed in Fig. 7 for simplicity.

3.3.1 Feature Extraction

As shown in Fig. 7a, we first apply a user-guided image segmentation technique to extract features in the raw BTF, the “holes” in this particular example. After the segmentation, the GUI will highlight the extracted features in red.

3.3.2 Corner Sampling

Fig. 7b presents the corner sampling process. These corner samples can be selected automatically or manually. To match the color tone of the background BTF with the color of the features, the user can optionally tune the hue and saturation of the background BTF. As the resultant color is basically a linear combination of SH coefficients, we can tune the color tone by tuning its SH coefficients directly. Note that background BTF refers to the BTF (usually low frequency) already synthesized on the BTF tiles.

3.3.3 Edge Synthesis

After generating the corner samples, we synthesize the edges. As depicted in Fig. 7c, we can interactively drag-and-drop previously segmented features onto the edge using the

Fig. 7. Demonstration of our user-controllable tile synthesis GUI: (a) feature extraction, (b) corner sampling, (c) interactive edge synthesis, and (d) interactive interior area synthesis.

GUI and apply a constrained texture synthesis to fill the blank area (highlighted in red). Since each synthesized edge is ultimately divided into two halves during the frame formation (see Fig. 4), we can match individual features even when they cross the edges of two tiles.

3.3.4 Interior Area Synthesis

Fig. 7d shows a resultant tile frame constructed using the edge generated in Fig. 7c. The lower half of the edge corresponds to the top edge of the frame. Similarly, we can also introduce features into the interior of a tile before applying the constrained texture synthesis to fill its interior blank area (highlighted in red).

With this GUI, we allow the user to take *full control* of the generation of the desired BTF in practical applications. Features can lie on the edges of tiles and are still matchable after the tiling. Sometimes, it could be hard to synthesize features with regular structures that are relatively large compared to the size of tiles, but with this GUI, we can constrain the features on edges and tiles and, thus, can preserve features originally in the raw BTF data.

Fig. 8. (a) A set of Wang tiles and (b) a valid Wang tiling.

4 TILING AND RENDERING

4.1 Tiling

Before dressing a geometry surface with the BTF tiles, we have to parameterize the geometry surface. Techniques concerning surface parameterization [3], [13], [17] were studied intensively in recent years. Examples include the shell map structure [16], [39] on various 3D models, as well as the PolyCube-Map method [49] for efficient texture mapping. In our current implementation, we adopt the PolyCube-Map method to install a quad-based structure on the input meshes. However, we have to emphasize that any low-distortion surface parameterization can be adopted in our tileable BTF framework.

After the surface parameterization, Wang tiling [9], [18], [46], [58] is employed to dress up the geometry surfaces with the matchable BTF tiles, and this useful tiling tool also helps to formulate the BTF tile arrangement onto the parameterized surfaces. As illustrated in Fig. 8, Wang tiles have color-coded edges, and the matching of edge color between tiles leads to a match in the texture pattern contained in the tiles. Thus, we can create a seamless texture pattern nonperiodically on the tiled region.

4.2 Rendering

Once we complete the above offline precomputation, the rendering of BTF tiles on geometry surfaces is straightforward. Fig. 9 illustrates the basic rendering mechanism: 1) the TBN transform [2], [38] and 2) the double SH reconstruction. The abbreviation TBN refers to the local tangent space on object surface, formed by tangent ("T"), binormal ("B"), and normal ("N").

Local illumination. To ensure that the SH coefficients are reusable for tiling different surfaces, they have to be encoded in the local tile space. Hence, the transformation of both light and viewing vectors from the object space to the local tile space is required for computing the local illumination. It is important to note that the tangents and binormals are not defined by the local curvature as in general TBN transform [2], [38]. Rather, we align them with the local surface parameterization grid so that the transformed light and viewing vectors conform to the coordinate system of the tile space as defined in the raw BTF samples. After the transformation, we look up the corresponding

Fig. 9. Transformation between tile space and object space.

quantized SH coefficients, unquantize them, perform double SH reconstruction (inverse transform) to obtain the reflectance, and compute the final pixel color.

Distant environment. To render the BTF-tiled objects illuminated in a distant environment, our system currently supports two approaches: 1) importance sampling [1] and 2) frequency-domain approach [43]. The importance sampling approach approximates the illumination of a distant environment using a limited number of directional lights, say, 200 lights. Efficient sampling algorithms have been proposed recently [1], [8], [54]. We used the Spherical Q^2 -Tree sampling technique [54] in generating the samples. For each sample (directional light), we render an image by local illumination. The final result is produced by summing the rendering results from multiple passes of such local illumination.

The frequency-domain approach first encodes the distant environment as an SH coefficient vector. The pixel color is computed by performing the inner product between the SH matrix (BRDF), $\vec{c}_\rho$, and the SH vector (distant environment), $\vec{c}_e$, in a frequency domain. However, a rotation, $\mathbf{R}$, on $\vec{c}_e$ has to be carried out at each pixel, as the local tile space BTF ($\vec{c}_\rho$) and the environment ($\vec{c}_e$) are SH-encoded in two different coordinate systems. Furthermore, note that the CLS we used actually encodes the BRDF in the hemispherical SH domain; the lower hemispheres of the basis functions are all zeros. An autocorrelation matrix, $\mathbf{A}$, is required to convert the full-sphere SH coefficients of distant environment before performing the inner product. Readers are referred to [32] for the mathematical details. In matrix form,

$$\text{the final pixel radiance } p = \vec{c}_\rho(\vec{v})^T [\mathbf{A}] [\mathbf{R}] \vec{c}_e, \quad (1)$$

where $\vec{v}$ is the viewing direction corresponding to the pixel in tile space, $\vec{c}_\rho(\vec{v})$ is a k_l -dimensional vector reconstructed given the current viewing direction $\vec{v}$, and $\mathbf{A}$ is a $k_l \times k_l$ matrix with elements

$$a_{ij} = \int_{\Omega_H} y_i(\vec{s}) y_j(\vec{s}) d\vec{s} \quad (2)$$

for the integral of two SH basis functions over the hemisphere Ω_H . This matrix of integrals can be precomputed numerically (see Appendix for the definition of y_i). $\mathbf{R}$ is a $k_l \times k_l$ matrix that rotates the environment coefficient vector $\vec{c}_e$ to align with $\vec{c}_\rho$ in local tile space. The elements of matrix $\mathbf{R}$ can be determined as described in [22]. However, a more efficient way to rotate $\vec{c}_e$ is to compute it analytically [27], [40]. In our current implementation, we compute the reflected radiance using this frequency-domain approach, and the rotation of $\vec{c}_e$ is computed analytically. In addition,

TABLE 1
Properties of Raw BTF Data Sets in Our Experiments

BTF	Original Spatial Image Resolution	Light Sampled	View Sampled	Total Size
FLOORTILE	800 × 800	81	81	12.6GB
IMPALLA	256 × 256	81	81	1.29GB
WRINKLES	128 × 128	50	60	147MB
HOLES	128 × 128	51	51	128MB
REACTDIFFUSE	128 × 128	81	81	321MB

our current implementation does not render the macroscale shadow due to the object. To account for this kind of shadow, the biscale radiance transfer [45] can be employed.

5 IMPLEMENTATION AND RESULTS

5.1 Data Sources

We have tested the proposed framework on both real and synthetic BTF data. The real data employed in our experiments is mainly from the Bonn BTF database [4], [41]: FLOORTILE and IMPALLA. These BTF data samples are captured with 81 sample lighting and 81 sample viewing directions, resulting in a set of total 6,561 sample images in RGB format. In addition to the Bonn BTF database, we also used two BTF data obtained from Microsoft Research Asia (MSRA) [35], [51], WRINKLES and HOLES, and another synthetic BTF data produced ourselves, REACTDIFFUSE. The total size of the BTF data sets ranges from 128 Mbytes to 12.6 Gbytes. Table 1 summarizes the properties of all raw BTF data used in our experiments.

5.2 Compact Data Representation

Since the plain BTF data samples could be too large to fit into the conventional PC memory, our data compression engine performs the double SH projection and the uniform quantization in a scan-line-wise fashion. This approach can greatly reduce the amount of disk I/O and optimize the data processing speed. Then, each coefficient is quantized to an 8-bit integer so that we can hold all quantized SH coefficients in memory afterward. Table 2 shows the timing and performance statistics of the compression including the double SH projection and the uniform quantization tested on a PC with Pentium IV 3.2-GHz CPU and 1 Gbyte of memory. Different numbers of SH coefficients are tested. Column “Timing” shows the time to perform double SH projection and uniform quantization. It increases as the raw BTF data size or the number of SH coefficients employed increases. “Compression ratio” is measured with respect to the raw BTF data size. It is mainly affected by the number of

TABLE 3
Statistics of Synthesized BTF Tile Sets

	Tile Resolution	Total Synthesis Time (min.)	Percentage of Referenced BTFels	GPU memory needed (in MB)
FLOORTILE	128 × 128	33	37.86%	186.10
IMPALLA	256 × 256	51	49.92%	61.36
WRINKLES	64 × 64	15	76.92%	23.64
HOLES	100 × 100	25	34.86%	10.72
REACTDIFFUSE	64 × 64	16	55.01%	16.91

SH coefficients employed and the original sampling rates along lighting and viewing.

5.3 Tile Synthesis

In our experiments, the time needed to arrange the BTF tiles on geometry surfaces is negligible compared to the time needed to synthesize the BTF tiles. Fortunately, we only need to perform the synthesis once in offline. Table 3 lists the total time for synthesizing each BTF tile set tested in our experiment (column “Total Synthesis Time”). The statistics are recorded on a PC with Pentium IV 3.2-GHz CPU and 1 Gbyte of memory.

After the BTF tile synthesis, not all BTF elements (BTFels) could be finally used in a tile set, whereas many other BTFels could be repeatedly used in different BTF tiles. Keeping all SH coefficients for each BTF tile is wasteful. To efficiently store the data, we construct the BTFel table, which stores only those referenced BTFels without duplication. Then, for each BTF tile, we only store a 2D array of the BTFel index pointing to elements in the BTFel table, instead of directly storing the full SH coefficients. This means that storing those synthesized BTF tiles in GPU memory can be highly efficient. The column “Percentage of Referenced BTFels” in Table 3 lists the percentage of BTFels referenced and, hence, stored for each synthesized BTF tile set in our experiments and, also, the amount of GPU memory needed to store the BTFels together with the uniform quantization coefficients. The ratio (the fourth column) is measured with respect to the total number of texels in the raw BTF data. In addition, note that all these BTF tile sets have 96 BTF tiles and use 25×25 lighting and viewing SH coefficients, except for the FLOORTILE data set, which has only 16×16 coefficients so that it can be fitted into the texture memory.

5.4 Results

To demonstrate the renderings of BTF-dressed surface under different lighting and viewing conditions, two kinds of lighting configurations are used in our experiments: local illumination and distant environment lighting. Figs. 10, 11, and 12 present the rendering results of BTF-dressed objects illuminated by a point light source and as viewed from different orientations. In Figs. 10, 11, and 12, the BTF tile sets shown are IMPALLA, HOLES, and WRINKLES, respectively. In each figure, the same BTF tile set is repeatedly used to seamlessly dress three objects, BUNNY (the top row), 3-HOLES (the middle row), and LAURANA (the bottom row). For each generated image, two boxed regions are blown up for inspection.

Fig. 13 shows the distant environment lit BUNNY. In the upper row, BUNNY is dressed with the BTF tile set WRINKLES and lit by GRACE. The lower row shows the BUNNY dressed with REACTDIFFUSE lit in GALILEO. Importance sampling approach is used in order to render

TABLE 2
Performance of Data Compression (Including Double SH Projection and Uniform Quantization)

Resolution × light × view	No. of Coefficients		Timing (h:m:s)	Compress. Ratio
	View(k_v)	Light(k_l)		
128 × 128 × 81 × 81 (321MB)	25	25	10 : 13	9.53%
	36	25	12 : 41	13.73%
256 × 256 × 81 × 81 (1.29GB)	25	25	17 : 47	9.53%
	36	25	20 : 51	13.73%
800 × 800 × 81 × 81 (12.60GB)	25	25	2 : 51 : 29	9.53%
	36	25	3 : 22 : 00	13.73%

Fig. 10. The BTF tile set: IMPALLA.

the macroscale shadow due to the object. In particular, we applied the Spherical Q^2 -Tree [54] to generate the samples (directions). For all the results in Fig. 13, 50 samples are used, and the corresponding 50 locally illuminated images are summed to generate the final result. The total time taken to render a 640×480 image are 3.2 minutes and 4.5 minutes for WRINKLES-dressed and REACTDIFFUSE-dressed BUNNY, respectively.

With the appearance-geometry decoupling, we can see that the same BTF tile set can be repeatedly used to dress up surfaces seamlessly even the geometries are substantially different.

5.5 Limitations

One major limitation of our method is that the BTF being synthesized must be more or less “isotropic” spatially. If the

Fig. 11. The BTF tile set: HOLES.

BTF exhibits an obvious anisotropic pattern, extra effort is needed to maintain the anisotropic behavior over the object surface. This is because our tiling approach only ensures the matching locally among tile edges.

Another restriction is due to the nature of the BTF. Unlike simple color textures (diffuse reflection) that are independent of viewing and lighting directions, there is always an intrinsic orientation associated with the BTF as the BTF is

acquired under a specific orientation (coordinate framework). Note that the BTF captures all frequency components including the specular component, as well as the diffuse component. During the synthesis, we have to match the pixel values together with the orientation (the spatial ordering of the pixels). Therefore, all synthesized BTF tiles must be laid on the tile surface with a consistent orientation,

Fig. 12. The BTF tile set: WRINKLES.

unlike the color texture tiles that can be randomly oriented, provided the edges are matched.

The proposed framework relies on a proper quad-based parameterization over the object surface. If the parameterized quads are not maintained with similar sizes or the quads are overdistorved, the final dressing may be malformed. This is mainly due to the fact that BTF tiles are synthesized in a single scale and in a square shape.

6 CONCLUSION

In conclusion, this paper presents a novel and modular framework for efficiently applying the BTF on geometry surfaces. Given M different surfaces and N different BTF data samples, if we apply conventional BTF approach, we need to perform the BTF synthesis $M \times N$ times so as to produce all different BTF dressings on the M different surfaces. In contrast, with the appearance-geometry

Fig. 13. Distant environment lighting: a BTF-dressed BUNNY illuminated by the high dynamic range (HDR) environments. Upper row: WRINKLES lit by GRACE. Lower row: REACTDIFFUSE lit by GALILEO.

decoupling, the proposed framework only needs to perform the synthesis process N times so as to generate N sets of BTF tiles corresponding to the N BTF inputs. Once these tile sets are constructed, we do not need to change them anymore. They can be repeatedly used to dress up M or more 3D models without retile a model or resynthesizing a tile set. In this way, the BTF becomes highly reusable. Game developers can stock a library of BTF tiles and conveniently apply them to dress up different models in a highly cost-effective manner.

To make this modular framework practical, we also introduce an original tile synthesis algorithm for synthesizing the BTF tiles. It divides the BTF tile synthesis into four substeps: corner sampling, edge synthesis, frame construction, and interior area synthesis. Such approach relaxes the constraints (by avoiding long cutting paths) and increases the choices of samples in order to maximize the chance of synthesizing seamless tiles for the high-dimensional BTF. A companion video and further information can be found at <http://www.cse.cuhk.edu.hk/~ttwong/papers/btftile/btftile.html>.

APPENDIX

CLS ESTIMATION FOR SH

Given M samples of a spherical function f , $\vec{\beta} = [f(\vec{s}_1), \dots, f(\vec{s}_M)]^T$, sampled at directions $\{\vec{s}_1, \dots, \vec{s}_M\}$, we want to

estimate an n -dimensional SH coefficient vector $\vec{c}$ that satisfies the following linear system:

$$\vec{\beta} = \mathbf{Y}\vec{c}, \quad (3)$$

where $y_i(\vec{s})$ is the i th SH basis function evaluated at direction $\vec{s}$ and

$$\mathbf{Y} = \begin{bmatrix} y_1(\vec{s}_1) & \cdots & y_n(\vec{s}_1) \\ \vdots & \ddots & \vdots \\ y_1(\vec{s}_M) & \cdots & y_n(\vec{s}_M) \end{bmatrix}.$$

We can set up the following CLS cost function:

$$J(\vec{c}) = \|\vec{\beta} - \mathbf{Y}\vec{c}\|^2 + \lambda(\vec{c}^T \vec{c} - E), \quad (4)$$

where E is the energy of function f and can be estimated from the sampled values of $f(\vec{s})$; $\lambda > 0$ is a parameter to search. Its constrained solution is given by

$$\vec{c} = (A + \lambda I_{n \times n})^{-1} \vec{b}, \quad (5)$$

where $A = \mathbf{Y}^T \mathbf{Y}$, $I_{n \times n}$ is an $n \times n$ identity matrix, and $\vec{b} = \mathbf{Y}^T \vec{\beta}$.

According to the constraint

$$\vec{c}^T \vec{c} = \vec{b}^T (A + \lambda I_{n \times n})^{-2} \vec{b} \leq E, \quad (6)$$

λ should satisfy

$$\vec{b}^T(A + \lambda I_{n \times n})^{-2}\vec{b} \leq E. \quad (7)$$

If we set $\lambda = 0$, the least square solution is obtained. If we set $\lambda \rightarrow \infty$, $\vec{c}$ is a zero vector. As λ increases, the norm of $\vec{c}$ decreases monotonically. The goal is to find out the smallest value of λ such that (7) is satisfied. As J can be proved to be an increasing function of λ , we use an iterative approach to determine λ by first assigning an initial value of λ and then using a simple binary search to estimate a suitable value of λ based on (7). For a detailed proof, readers are referred to [32].

ACKNOWLEDGMENTS

In this research work, the authors would like to thank Microsoft Research Asia (MSRA) and the University of Bonn for their high-quality BTF data, Paul Debevec for the high dynamic range (HDR) environment maps, Liang Wan and Guangyu Wang for helping them prepare the companion video, Marco Tarini of the Visual Computing Lab, Istituto di Scienza e Tecnologie dell'Informazione/Consiglio Nazionale delle Ricerche (ISTI/CNR), Pisa, for the PolyCube-Mapped models, and Yu-Wing Tai for his advice in implementing the graph-cut algorithm. This project is supported by the Research Grants Council of the Hong Kong Special Administrative Region, under Research Grants Council (RGC) Earmarked Grants (Project HKUST612706 and CUHK416806) and is affiliated with the Chinese University of Hong Kong (CUHK) Virtual Reality, Visualization and Imaging Research Centre, as well as the Microsoft-CUHK Joint Laboratory for Human-Centric Computing and Interface Technologies.

REFERENCES

- [1] S. Agarwal, R. Ramamoorthi, S. Belongie, and H.W. Jensen, "Structured Importance Sampling of Environment Maps," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 605-612, 2003.
- [2] J.F. Blinn, "Simulation of Wrinkled Surfaces," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '03)*, vol. 12, pp. 286-292, Aug. 1978.
- [3] I. Boier-Martin, H. Rushmeier, and J. Jin, "Parameterization of Triangle Meshes over Quadrilateral Domains," *Proc. 2004 Eurographics/ACM SIGGRAPH Symp. Geometry Processing*, pp. 193-203, 2004.
- [4] *Bonn BTF Database*, Univ. of Bonn, <http://btf.cs.uni-bonn.de>, 2003.
- [5] B. Cabral, N. Max, and R. Springmeyer, "Bidirectional Reflection Functions from Surface Bump Maps," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '87)*, vol. 21, pp. 273-281, July 1987.
- [6] Y. Chen, Y. Xu, B. Guo, and H.-Y. Shum, "Modeling and Rendering of Realistic Feathers," *ACM Trans. Graphics*, vol. 21, no. 3, pp. 630-636, 2002.
- [7] S. Chenney, "Flow Tiles," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 233-242, 2004.
- [8] J. Cohen and P. Debevec, *LightGen HDRShop Plugin*, <http://gl.ict.usc.edu/HDRShop/lightgen/lightgen.html>, 2001.
- [9] 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.
- [10] O.G. Cula and K.J. Dana, "Compact Representation of Bidirectional Texture Functions," *Proc. Conf. Computer Vision and Pattern Recognition (CVPR '01)*, pp. 1041-1047, 2001.
- [11] CURET, *Columbia-Utrecht Reflectance and Texture*, <http://www.cs.columbia.edu/CAVE/software/curet/index.php>, 1999.
- [12] 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.
- [13] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '95)*, pp. 173-182, 1995.
- [14] A.A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '01)*, pp. 341-346, 2001.
- [15] A.A. Efros and T.K. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 1033-1038, 1999.
- [16] G. Elber, "Geometric Texture Modeling," *IEEE Computer Graphics and Applications*, vol. 25, no. 4, pp. 66-76, July/Aug. 2005.
- [17] M.S. Floater and K. Hormann, "Surface Parameterization: A Tutorial and Survey," *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S. Floater, and M.A. Sabin, eds. Springer, pp. 157-186, 2005.
- [18] 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.
- [19] 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.
- [20] B. Van Ginneken, J.J. Koenderink, and K.J. Dana, "Texture Histograms as a Function of Irradiation and Viewing Direction," *Int'l J. Computer Vision*, vol. 31, no. 2-3, pp. 169-184, 1999.
- [21] B. Grünbaum and G.C. Shephard, *Tilings and Patterns*. W.H. Freeman, 1986.
- [22] R. Green, "Spherical Harmonic Lighting: The Gritty Details," *Proc. Game Developer Conf. (GDC '03)*, Mar. 2003.
- [23] J.Y. Han and K. Perlin, "Measuring Bidirectional Texture Reflectance with a Kaleidoscope," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 741-748, 2003.
- [24] P. Hanrahan and W. Krueger, "Reflection from Layered Surfaces Due to Subsurface Scattering," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '93)*, pp. 165-174, 1993.
- [25] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin, "Image Analogies," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '01)*, pp. 327-340, Aug. 2001.
- [26] P.-M. Ho, T.-T. Wong, and C.-S. Leung, "Compressing the Illumination-Adjustable Images with Principal Component Analysis," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 355-364, Mar. 2005.
- [27] J. Ivanić and K. Ruedenberg, "Rotation Matrices for Real Spherical Harmonics. Direct Determination by Recursion," *J. Physical Chemistry*, vol. 100, no. 15, pp. 6342-6347, 1999.
- [28] J. Kopf, D. Cohen-Or, O. Deussen, and D. Lischinski, "Recursive Wang Tiles for Real-Time Blue Noise," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 509-518, 2006.
- [29] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 277-286, 2003.
- [30] A. Lagae and P. Dutré, "A Procedural Object Distribution Function," *ACM Trans. Graphics*, vol. 24, no. 4, pp. 1442-1461, Oct. 2005.
- [31] A. Lagae and P. Dutré, "An Alternative for Wang Tiles: Colored Edges versus Colored Corners," *ACM Trans. Graphics*, vol. 25, no. 4, pp. 1442-1459, Oct. 2006.
- [32] 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.
- [33] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearance of Materials Using Three-Dimensional Textons," *Int'l J. Computer Vision*, vol. 43, no. 1, pp. 29-44, 2001.
- [34] X. Liu, Y. Hu, J. Zhang, X. Tong, B. Guo, and H.-Y. Shum, "Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces," *IEEE Trans. Visualization and Computer Graphics*, vol. 10, no. 3, pp. 278-289, 2004.
- [35] X. Liu, Y. Yu, and H.-Y. Shum, "Synthesizing Bidirectional Texture Functions for Real-World Surfaces," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '01)*, pp. 97-106, 2001.
- [36] G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein, "Acquisition, Synthesis and Rendering of Bidirectional Texture Functions," *Computer Graphics Forum*, vol. 24, no. 1, pp. 83-109, Mar. 2005.
- [37] A. Nealen and M. Alexa, "Hybrid Texture Synthesis," *Proc. Eurographics Symp. Rendering (EGSR '03)*, pp. 97-105, 2003.

- [38] M. Peercy, J. Airey, and B. Cabral, "Efficient Bump Mapping Hardware," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '97)*, pp. 303-306, 1997.
- [39] S.D. Porumbescu, B. Budge, L. Feng, and K.I. Joy, "Shell Maps," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 626-633, 2005.
- [40] D.W. Ritchie and G.J.L. Kemp, "Fast Computation, Rotation and Comparison of Low Resolution Spherical Harmonic Molecular Surfaces," *J. Computational Chemistry*, vol. 20, no. 4, pp. 383-395, 1999.
- [41] M. Sattler, R. Sarlette, and R. Klein, "Efficient and Realistic Visualization of Cloth," *Proc. Eurographics Symp. Rendering (EGSR '03)*, pp. 167-177, <http://btf.cs.uni-bonn.de/download.html>, 2003.
- [42] J. Shade, M.F. Cohen, and D.P. Mitchell, "Tiling Layered Depth Images," technical report, Univ. of Washington, Seattle, 2002.
- [43] F.X. Sillion, J.R. Arvo, S.H. Westin, and D.P. Greenberg, "A Global Illumination Solution for General Reflectance Distributions," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '91)*, vol. 25, pp. 187-196, July 1991.
- [44] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder, "Clustered Principal Components for Precomputed Radiance Transfer," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 382-391, 2003.
- [45] P.-P. Sloan, X. Liu, H.-Y. Shum, and J. Snyder, "Bi-Scale Radiance Transfer," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 370-375, 2003.
- [46] J. Stam, "Aperiodic Texture Mapping," Technical Report R046, European Research Consortium for Informatics and Math., 1997.
- [47] P.-H. Suen and G. Healey, "Analyzing the Bidirectional Texture Function," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, p. 753, 1998.
- [48] F. Suykens, K. vom Berge, A. Lagae, and P. Dutré, "Interactive Rendering with Bidirectional Texture Functions," *Computer Graphics Forum*, vol. 22, no. 3, Sept. 2003.
- [49] M. Tarini, K. Hormann, P. Cignoni, and C. Montani, "PolyCube-Maps," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 853-860, 2004.
- [50] X. Tong, J. Wang, S. Lin, B. Guo, and H.-Y. Shum, "Modeling and Rendering of Quasi-Homogeneous Materials," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 1054-1061, 2005.
- [51] 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.
- [52] G. Turk, "Texture Synthesis on Surfaces," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '01)*, pp. 347-354, 2001.
- [53] M. Alex, O. Vasilescu, and D. Terzopoulos, "Tensortextures: Multilinear Image-Based Rendering," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 336-342, 2004.
- [54] L. Wan, T.-T. Wong, and C.-S. Leung, "Spherical Q2-Tree for Sampling Dynamic Environment Sequences," *Proc. Eurographics Symp. Rendering (EGSR '05)*, pp. 21-30, June 2005.
- [55] H. Wang, "Proving Theorems by Pattern Recognition II," *Bell Systems Technical J.*, vol. 40, pp. 1-42, 1961.
- [56] H. Wang, Q. Wu, L. Shi, Y. Yu, and N. Ahuja, "Out-of-Core Tensor Approximation of Multi-Dimensional Matrices of Visual Data," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 527-535, 2005.
- [57] L. Wang, W. Wang, J. Dorsey, X. Yang, B. Guo, and H.-Y. Shum, "Real-Time Rendering of Plant Leaves," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 712-719, 2005.
- [58] L.-Y. Wei, "Tile-Based Texture Mapping on Graphics Hardware," *Proc. SIGGRAPH/Eurographics Conf. Graphics Hardware*, pp. 55-63, 2004.
- [59] L.-Y. Wei and M. Levoy, "Texture Synthesis over Arbitrary Manifold Surfaces," *Proc. Int'l Conf. Computer Graphics (SIGGRAPH '01)*, pp. 355-360, 2001.
- [60] T.-T. Wong, P.-A. Heng, S.-H. Or, and W.-Y. Ng, "Image-Based Rendering with Controllable Illumination," *Proc. Eighth Eurographics Workshop Rendering (Rendering Techniques '97)*, pp. 13-22, June 1997.
- [61] T.-T. Wong and C.-S. Leung, "Compression of Illumination-Adjustable Images," *IEEE Trans. Circuits and Systems for Video Technology*, special issue on image-based modeling, rendering and animation, vol. 13, no. 11, pp. 1107-1118, Nov. 2003.
- [62] S. Zelinka and M. Garland, "Jump Map-Based Interactive Texture Synthesis," *ACM Trans. Graphics*, vol. 23, no. 4, pp. 930-962, 2004.
- [63] 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.

Man-Kang Leung received the BSc and MPhil degrees in computer science from the Hong Kong University of Science and Technology in 2004 and 2006, respectively. He is now a research assistant working in the computer graphics group. His research interests include texture synthesis, tiling, the BTF, and geometric modeling.

Wai-Man Pang received the BSc and the MPhil degrees in computer science from the Chinese University of Hong Kong. He is currently a PhD candidate in the Department of Computer Science and Engineering, Chinese University of Hong Kong. His research interests include image-based rendering, graphics processing unit (GPU) programming, nonphotorealistic rendering, and physically-based deformation. He is a student member of the IEEE.

Chi-Wing Fu received the BSc and MPhil degrees in computer science from the Chinese University of Hong Kong in 1997 and 1999, respectively, and the PhD degree in computer science from Indiana University at Bloomington in 2003. He is now a visiting assistant professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests include image-based modeling and rendering, texture synthesis, medical visualization, and visualization and navigation in large-scale astrophysical environments. He is a member of the IEEE, the IEEE Computer Society, and ACM SIGGRAPH.

Tien-Tsin Wong received the BSc, MPhil, and PhD degrees in computer science from the Chinese University of Hong Kong in 1992, 1994, and 1998, respectively. Currently, he is a professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong. His main research interests include computer graphics, including image-based rendering, natural phenomena modeling, and multimedia data compression. He received the *IEEE Transactions on Multimedia* Prize Paper Award 2005 and the Young Researcher Award in 2004. He is a member of the IEEE Computer Society.

Pheng-Ann Heng (M'92, SM'06) received the BSc degree from the National University of Singapore in 1985 and the MSc degree in computer science, the MA degree in applied mathematics, and the PhD degree in computer science, all from Indiana University, Bloomington, in 1987, 1988, and 1992, respectively. Currently, he is a professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong (CUHK), Shatin. In 1999, he set up the Virtual Reality, Visualization, and Imaging Research Centre at CUHK and serves as the director of the center. He is also the director of the Human-Computer Interaction Research Centre, Shenzhen Institute of Advanced Integration Technology, Chinese Academy of Science/Chinese University of Hong Kong. His research interests include virtual reality applications in medicine, visualization, medical imaging, human-computer interface, rendering and modeling, interactive graphics, and animation. He is a senior member of the IEEE.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.