The Visual Computer manuscript No.
(will be inserted by the editor)

Jiaping Wang · Xin Tong · John Snyder · Yanyun Chen · Baining Guo · Heung-Yeung Shum

Capturing and Rendering Geometry Details for BTF-mapped Surfaces

Abstract Bidirectional texture functions or BTFs accurately model reflectance variation at a fine (meso-) scale as a function of lighting and viewing direction. BTFs also capture view-dependent visibility variation, also called masking or parallax, but only within surface contours. Mesostructure detail is neglected at silhouettes, so BTF-mapped objects retain the coarse shape of the underlying model.

We augment BTF rendering to obtain approximate meso-scale silhouettes. Our new representation, the 4D mesostructure distance function (MDF), tabulates the displacement from a reference frame where a ray first intersects the meso-scale geometry beneath, as a function of ray direction and ray position along that reference plane. Given an MDF, the mesostructure silhouette can be rendered with a per-pixel depth peeling process on graphics hardware, while shading and local parallax is handled by the BTF. Our approach allows real-time rendering, handles complex, non-height-field mesostructure, requires that no additional geometry to be sent to the rasterizer other than the mesh triangles, is more compact than textured visibility representations used previously, and for the first time can be easily measured from physical samples. We also adapt the algorithm to capture detailed shadows cast both by and onto BTF-mapped surfaces. We demonstrate the efficiency of our algorithm on a variety of BTF data, including real data acquired using our BTF-MDF measurement system.

Keywords Bidirectional texture functions · Reflectance and shading models · Rendering · Shadow algorithms · Texture mapping

Jiaping Wang†
Institute of Computing Technology, Chinese Academy of Science
Graduated School of Chinese Academy of Science
E-mail: e_boris2002@hotmail.com

Xin Tong Yanyun Chen Baining Guo Heung-Yeung Shum
Microsoft Research Asia
E-mail: {xtong,yachen,bainguo,hshum}@microsoft.com

John Snyder
Microsoft Research
E-mail: johnsny@microsoft.com

†This work was done while Jiaping Wang was visiting Microsoft Research Asia.

1 Introduction

Real-world textures arise from both spatially-varying surface reflectance and mesostructure wich represents fine-scale geometric details [14]. The 6D bidirectional texture function (BTF) captures these variations [6] and has been exploited to measure real materials and map their properties onto arbitrary geometric models [27, 25, 26, 23]. Rendering BTF-mapped surfaces yields rich visual effects such as fine-scale shading, shadows, inter-reflections, and occlusions, but omits detailed silhouettes. This is because the BTF only provides radiance measurements as a function of lighting and viewing direction, leaving meso-structure visibility implicit. Silhouettes are thus identical to that of the underlying mapped surface, as shown in Figure 1a.

To solve this problem, we build on recent methods that explicitly tabulate local visibility. The view-dependent displacement map (VDM) [29] precomputes the mesostructure visibility as a function of texture position, view direction, and curvature. The result is stored as a 5D table and allows real-time rendering with graphics hardware. The generalized displacement map (GDM) [30] also stores a 5D table of displacements, but as a function of ray direction and 3D position in a volume. Both representations rely on synthetic visibility data, and it is not clear how either can be acquired from real materials. Moreover, the 5D table required strains the hardware memory system, making the approach less practical. Most fundamentally, neither method was combined with BTF rendering, but instead used traditional normal-mapped shading and two-pass shadowing. Using a BTF efficiently provides local shadowing without a shadow buffer, as well as other more general effects such as diffuse inter-reflections and spatially-varying reflectance, which are impractical with traditional shading techniques.

To add silhouette visibility to BTF rendering, we propose a new, image-based method which represents mesostructure as a 4D *mesostructure distance function* (MDF). It is used as a companion function for a given BTF and records, for each BTF viewing direction, the displacement along that direction of the mesostructure from the reference plane. Now we can determine whether the mesostructure is visible at the sil-

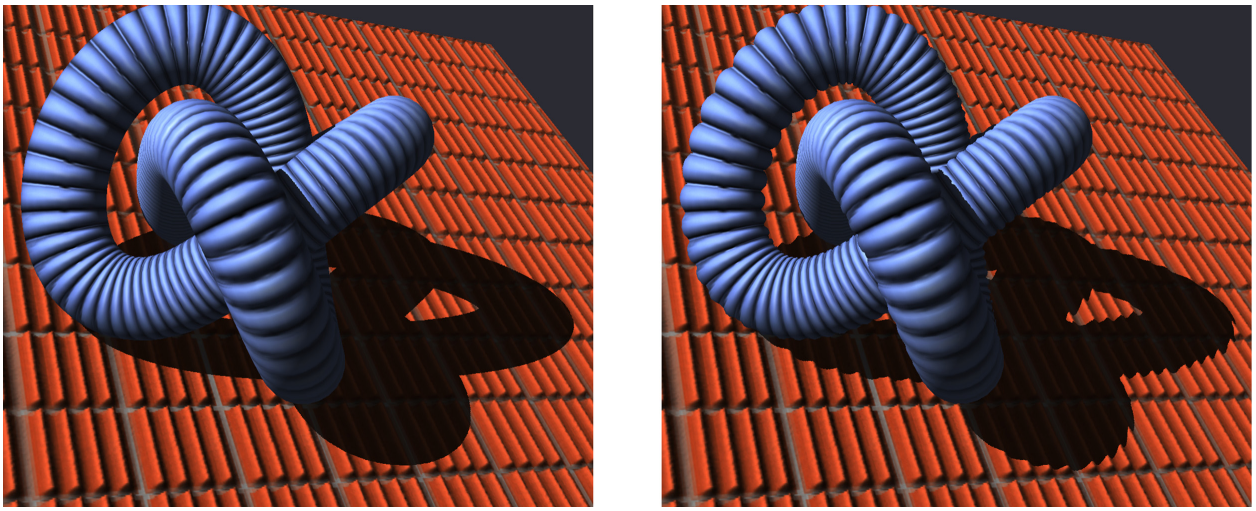

Fig. 1 (a) Existing BTF rendering techniques produce unrealistic silhouettes and shadow boundaries. (b) Our technique generates detailed silhouettes and shadow boundaries for BTF-mapped objects.

houette by rendering the front surface and the corresponding back surfaces and testing whether the depth between them exceeds the sum of the displacements indexed from the MDF at these two surfaces. Since object surfaces self-occlude, we use a multi-pass depth peeling approach to ensure that we can get the front surface and its corresponding back surfaces in silhouette rendering. Moreover, depth peeling ensures that we see through closer layers where the view ray misses the mesostructure to further ones where it may hit. The result provides visually important silhouette effects and can be extended to shadows, as shown in Figure 1b.

The MDF can be seen as a simplified VDM without its curvature parameter. However, this simplification is not as straightforward as it may appear. The VDM rendering method *relies* on a curvature parameter to obtain silhouette visibility effects. In other words, setting the curvature parameter to 0 and using the VDM rendering method [29] results in silhouettes from the underlying coarse model, just as with BTF rendering. It is only by using both our MDF representation and rendering method that we obtain meso-scale silhouette effects.

Our primary contribution is to add detailed silhouettes to BTF rendering. We show how our MDF representation can be obtained by measuring real samples at the same time the BTF is acquired. We also demonstrate realistic shadowing as well as silhoutte effects, including for the first time shadowing *onto* surfaces mapped with meso-scale geometry. Our results demonstrate an efficient solution to simulate all significant visual effects caused by geometric details on BTF-mapped surfaces.

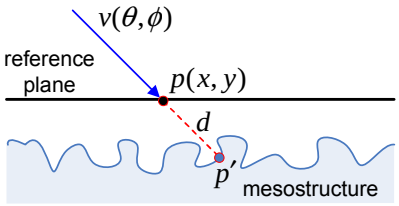

Fig. 2 Definition of the MDF

2 Related Work

A number of techniques measure BTFs from real materials [6,5,8,23,9,31]. Such BTFs can be mapped or synthesized onto geometric models (e.g., [16,27]). BTFs have also been used for rendering complex, non-height-field mesostructures such as fabric weaves [23] and fur [8]. The polynomial texture map [17] may be regarded as a special form of BTF with a fixed view.

Despite the high-dimensional data required, BTF-mapped surfaces can be rendered in real time [26,23,20,15,28]. These methods decompose the 6D BTF data into a combination of lower dimensional textures easily loaded and rendered on graphics hardware. [25] applies the pre-computed radiance transfer (PRT) technique to BTF-mapped surfaces. All these approaches ignore the mesostructure silhouette.

For height-field mesostructure [1,3], many techniques exist to achieve shading and parallax effects.

Horizon mapping generates meso-scale shadows on bump maps [19]. It can be accelerated using graphics hardware [24], and the representation can be captured from real samples [22]. Precomputed visibility [10] also renders meso-scale shadows and interreflections on bump-mapped surfaces. These methods neglect visibility effects entirely (e.g., parallax from bumps) both within surface contours as well as at silhouettes. They also lack the generality and realism of full BTF shading.

The parallax map [12] captures local parallax of a height field in real time. Most recently, [21] propose a technique to render the self-occlusions, interpenetrations and shadows of relief-mapped surfaces in real time. As is the case with BTF techniques, the underlying object's silhouette is unchanged in these approaches. Shading is computed traditionally; i.e., is non-BTF.

Hirche et al. [11] propose an algorithm to render a displaced surface on a per-pixel basis. Each triangle of an underlying mesh is extruded to form a prism. The intersection of the ray and extruded triangle is computed at each pixel of the prism faces. (A similar approach is also employed by the GDM method and must also rasterize prism faces rather than just mesh triangles.) Although the technique can render the detailed silhouette of a displaced surface in real time, it is limited to height-field mesostructures and so precludes many real-world examples including the grass and weave textures in our results. We also demonstrate an acquisition system that measures both the BTF and MDF (visibility) simultaneously, and a more realistic, BTF-based rendering method.

3 MDF Measurement

For a given BTF, the companion MDF records the distance of the mesostructure surface from a reference surface along a BTF viewing direction. As illustrated in Figure 2, the MDF is a 4D function $d = D(x,y,\theta,\phi)$, where (x,y) is the texture coordinate and the unit vector (θ,ϕ) is a sample viewing direction of the BTF expressed in spherical coordinates. For a given direction (θ,ϕ), each reference surface point $p = p(x,y)$ projects to a mesostructure point p' along the direction (θ,ϕ), and $D(x,y,\theta,\phi)$ is defined to be the distance from p to p'. To facilitate the mesostructure visibility computation, we define both BTF and MDF on a reference surface that lies above the top of mesostructure details.

Our MDF measurement device is a simple laser scan unit consisting of a laser source and a controller built with LEGO MindStorms Robotics Invention System 2.0. The laser scan unit is added to an existing BTF measurement system to measure a companion MDF for each captured BTF. Specifically, we generate a displacement image $D_v(x,y) = D(x,y,v)$ for every viewing direction v of the BTF.

3.1 System Setup

MDF measurement is closely coupled with BTF measurement. As illustrated in Figure 3 (a), our BTF measurement device mainly consists of a turntable and two concentric arcs, one for light sources and the other for cameras. The stationary camera arc shown on the left holds eight Dragonfly cameras, each providing 1024×768 24-bit color images at 15 Hz. The cameras are calibrated as in [32]. The light arc on the right rotates horizontally, driven by a stepping motor. Eight halogen lamps are mounted on the light arc. The light sources are manually calibrated as in [2]. The turntable in the middle holds material samples to be measured and is driven by another stepping motor. All the lamps, cameras, and stepping motors are controlled by a PC.

BTF capture is a fully automatic process controlled by the PC. After rotating the light arc and turntable to their desired sampling positions, the PC sequentially turns on/off the lamps. With each lamp being turned on, every camera records an image. The captured images are rectified and then clipped to form the output BTF data.

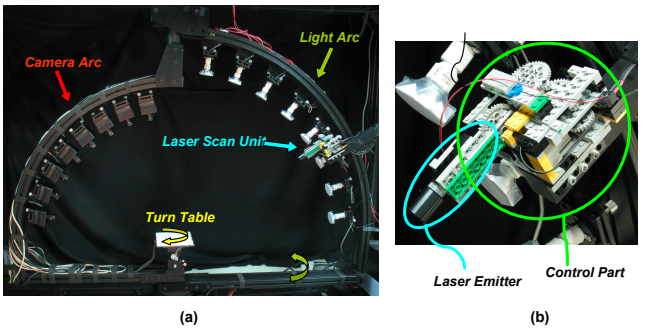

Fig. 3 (a) Our BTF-MDF measurement system. (b) The laser scan unit.

Figure 3 (b) shows the laser scan unit, which contains a laser strip generator and a controller. A miniature structured light diode laser serves as the light source, with a 60-degree line lens placed in front of the laser to project a laser strip on the turntable. The controller includes a robust gear-driven system that rotates the laser emitter vertically so that the laser strip scans across the turntable. We adjust the line lens direction to ensure that the laser scanning direction on the turntable is orthogonal to the laser strip. The laser unit is mounted on the light arc as shown in Figure 3 (a).

In principle we could build a separate laser range scanner for measuring the MDF. By combining MDF with BTF acquisition, we use the same cameras for measuring both the BTF and MDF and avoid cross-registration problems. Our laser unit is easy to build and add to an existing BTF device.

3.2 Capturing Process

The MDF measurement process involves three steps. First we project laser strips onto the material sample to be measured. As a laser strip scans across the sample, each camera records an image sequence. For a given camera with viewing direction v, the laser strip is often partly occluded from the camera by the mesostructure being measured. For this reason it is necessary to scan the material sample using laser strips projected from multiple source locations so that each mesostructure point visible from the camera is scanned at least once. Thus for each v we capture a number of laser scan image sequences, one for each laser source location. Then we apply space-time analysis to recover a displacement image for each captured image sequence. Finally, for every viewing direction v we merge all displacement images of v to generate $D_v(x,y)$.

Scan Image Sequences: For each viewing direction v, we scan the material sample with the laser from 26 source locations as illustrated in Figure 4. Since the laser emitter is far away from the turntable, we regard all the laser planes (a laser plane is the plane containing the laser strip and laser source point) in one scan as being parallel. Interreflections on the mesostructure surface can produce noise in the laser scan images. We resolve this problem by using a quick shutter speed so that the interreflections are under exposed and can be filtered out by post-processing. Alternatively we can

spray the material sample with matte paint before MDF measurement.

Before recording laser scan images for each laser source location, we first compute the angle between the laser plane and the reference plane (the turntable plane). To do this we project the laser strip onto two planes with different heights and calculate the offset of the laser strip in the two planes from two recorded laser scan images. From this offset we derive the angle between the laser plane and the reference plane. Once the angle is found, laser scan images of the material sample are recorded automatically. Specifically, we rotate the light arc and turntable to the desired sampling position. As the laser strip scans across the material sample, an image sequence is recorded by each BTF device camera at 7.5 Hz.

An MDF capturing session produces a set of laser scan image sequences. The image sequence for laser source direction l and viewing direction v is $\{I_{v,l}(x,y,t_i)\}$.

Initial Displacement Images: From each laser scan image sequence $\{I_{v,l}(x,y,t_i)\}$, we compute the displacement image $D_{v,l}(x,y)$ using space-time analysis [4]. A similar method has been used for shadow scanning [2].

Before the space-time analysis, we compute the laser plane for every image $I_{v,l}(x,y,t_i)$ in two steps. The first step is to detect the laser line in the image. Using the bounding box of the material sample, we divide $I_{v,l}(x,y,t_i)$ into two parts: the reference region showing only the turntable and the sample region where the material sample is imaged. The bounding box is manually specified on the reference plane before capturing. The laser strip should be a line in the reference region since it is flat. As illustrated in Figure 5a, to find this line, we search the reference region for candidate pixels on the line along the laser scan direction. Since the candidate pixels are by far the brightest pixels in the image, it is easy to reject outliers using a prescribed threshold. For noise reduction, we Gaussian-filter the image before the search. After finding all candidate pixels, we fit a line to them. In the second step, we calculate the laser plane for the image $I_{v,l}(x,y,t_i)$ by projecting the laser line from $I_{v,l}(x,y,t_i)$ to the 3D reference plane.

Once the laser planes of all images are found, the space-time analysis proceeds as follows. As illustrated in Figure 5b, for each pixel (x_s,y_s) in the sample region, we search the temporal sequence $\{I_{v,l}(x_s,y_s,t_i)\}$ to find the pixel (x_s,y_s,t_{i_0}) that has the brightest $I_{v,l}(x_s,y_s,t_{i_0})$. If $I_{v,l}(x_s,y_s,t_{i_0})$ is greater

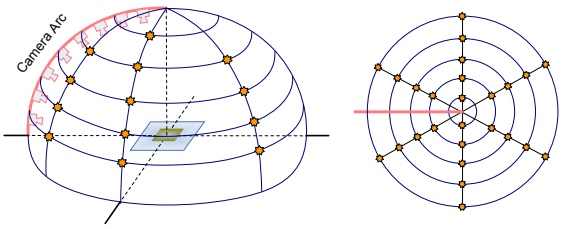

Fig. 4 The laser source locations are marked by the orange dots. Currently we simply re-mount the laser unit on the light arc manually (5 times per capturing session). Alternatively we can mount several identical units along the light arc, or use a mobile unit.

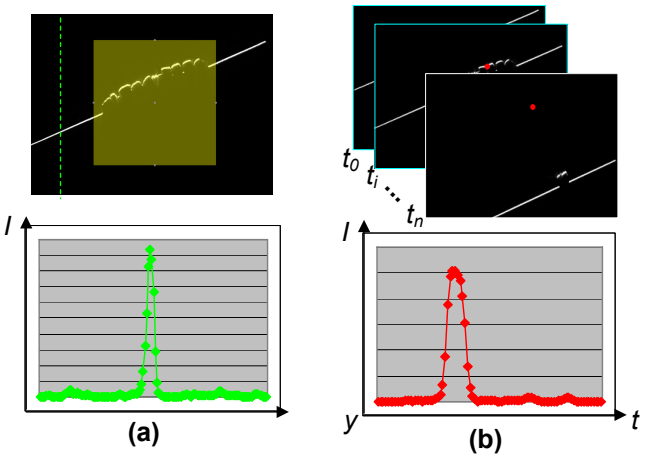

Fig. 5 The MDF capturing process. (a) Laser line detection. The bottom green curve illustrates the intensity of the pixels on the vertical line. The peak point corresponds to the candidate pixel on the laser line. (b) Space-time analysis. The bottom red curve illustrates the intensity variation of a pixel in whole scanning sequence. The peak point corresponds to the time at which this pixel images a mesostructure point.

than the prescribed threshold, then the pixel (x_s,y_s,t_{i_0}) images a mesostructure point p. We find p by projecting the pixel (x_s,y_s,t_{i_0}) onto the laser plane of the image $I_{v,l}(x,y,t_{i_0})$. We then compute the displacement of p from the reference plane and record the result in $D_{v,l}(x_s,y_s)$. If the $I_{v,l}(x_s,y_s,t_{i_0})$ is less than the prescribed threshold, the pixel (x_s,y_s,t_{i_0}) corresponds to an occluded mesostructure point. In this case, we set $D_{v,l}(x_s,y_s) = -1$.

Merging Displacement Images: For every viewing direction v, we merge all displacement images $D_{v,l}(x,y)$ to generate $D_v(x,y)$.

$$D_v(x,y) = \frac{1}{\sum w_i} \sum w_i D_{v,l_i}(x,y) \qquad (D_{v,l_i}(x,y) \neq -1)$$

, where the w_i is the weight of the scan result with laser direction l_i. Based on the error analysis in [2], we compute the weight for each scan $D_{v,l}(x,y)$ as the volume of the tetrahedron formed by the laser source position, the camera location, and two points on the laser line passing through the turntable center.

4 Rendering BTF-mapped Surfaces with MDF

After the BTF is mapped or synthesized onto arbitrary surfaces [27], the associated MDF texture can be mapped onto the surface with the same texture coordinates. The MDF data mapped onto surface is then used for rendering the mesostructure silhouette and mesostructure shadow silhouette, and also for enhancing shadow casting for BTF-mapped surface. All these computations can be performed in the pixel shader and accelerated with graphics hardware.

4.1 Visibility Computation

With the MDF, we can render mesostructure silhouettes as well as mesostructure shadow silhouettes. These two types

of silhouettes are intimately related in that both are results of visibility changes caused by mesostructures. For silhouettes the visibility is that from the viewpoint; for shadow silhouettes the visibility is that from the light source. The visibility computation algorithm discussed here is limited to closed objects.

Figure 6 illustrates the algorithm for determining mesostructure silhouettes. For simplicity we consider the silhouette determination problem in 2D (we can reduce the original 3D problem to 2D by performing silhouette determination for each scanline of the rendered image). Consider the viewing ray r_0 that enters the object surface at point p_f on a front face and leaves at point p_b on back face. With MDF value $D(x_f, y_f, v_f)$ on p_f, the offset from p_f to the mesostructure surface along r_0 is computed as $d_f = D(x_f, y_f, v_f)s_f$, where s_f is the texture scale at p_f. (x_f, y_f) is the texture coordinate at p_f, and v_f is the viewing direction of r_0 computed in the local coordinate frame at p_f. Similarly, the distance from point p_b to the mesostructure surface is $d_b = D(x_b, y_b, v_b)s_b$. As illustrated in Figure 6, if $d_f + d_b \leq ||p_f - p_b||$, the ray will intersect the mapped mesostructure at some point. Otherwise, the ray will pass through the mesostructure without intersection. If the ray intersects the object multiple times, we do the same computation for each pair of p_f and p_b to determine the visibility of the mesostructure mapped on the surface.

To handle the mesostructure silhouette with graphics hardware, we use the depth peeling technique [18,7] to find p_f and p_b for each interval inside the objects. As illustrated in Figure 7, before the peeling we initialize the depth buffer and stencil buffer. We also need two extra depth buffers for peeling, *Zpeel_now* and *Zpeel_next*, which store the minimum depth for the current and next peeling pass, respectively. In our algorithm, we execute the depth peeling from front to back. In each peeling pass, we first render the back faces. The *Zpeel_now* and the depth buffer Z work together to obtain the pixels p_b of the back faces in the current peel layer, whose depths are stored in *Zpeel_next*. We then offset their depths with the MDF value and store the result $(z_{p_b} - d_b)$ in depth buffer. After that, we disable depth buffer writing and render all front faces. For pixels p_f that pass the peeling depth test, we compare their offset depths $z_{p_f} + d_f$ with $z_{p_b} - d_b$ stored in the depth buffer. If $z_{p_f} + d_f \leq z_{p_b} - d_b$, the ray intersects with the mesostructure detail between the p_f and p_b. Otherwise, the ray passes through this peeling layer without intersection. For pixels whose rays intersect with mesostructure details, we shade p_f with BTF data according to the viewing and lighting conditions. After that,

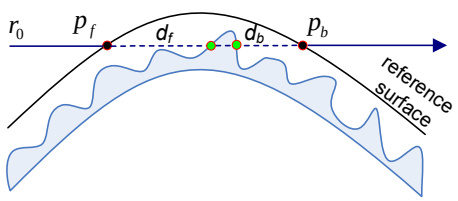

Fig. 6 Determining mesostructure silhouettes using the MDF.

```
Initial stencil buffer;
Enable stencil test;
Enable depth test;
Set Zpeel_next to 0.0;
For each peeling pass
    Set depth buffer Z to far plane;
    Zpeel_now = Zpeel_next;
    Enable depth buffer Z writing;
    Render all back faces;
    For all pixels p_b of back faces
        If z(p_b) ≥ Zpeel_now(p_b) and z(p_b) < Z(p_b)
        Zpeel_next(p_b) = z(p_b);
        Compute d(p_b) with MDF;
        Z(p_b) = z(p_b) - d(p_b);
    Disable depth buffer Z writing;
    Render all front faces;
    For all pixels p_f of front faces
        If z(p_f) > Zpeel_now(p_f) and z(p_f) < Z(p_f)
        Compute the d(p_f) with MDF;
        If z(p_f) + d(p_f) ≤ Z(p_f) and p_f is not masked
            Shade p_f with BTF;
            Mask p_f with stencil;
```

Fig. 7 Pseudo code of silhouette rendering.

we mask the shaded pixel using the stencil buffer so that it will not be corrupted with the surfaces in the following peeling layers. Then we swap the peeling depth buffer and do the next peeling pass. In theory, the peeling operation should be continued until no pixels on the screen are updated. In our current implementation, the number of the peeling passes for each scene is specified by the user.

Determining mesostructure shadow silhouettes is done in the same way except we render the scene for the lighting direction. Instead of shading the surface pixels p_f with BTF, we render their depths to the shadow map buffer.

4.2 Shadow Casting on BTF-mapped Surfaces

To render the BTF-mapped surface realistically, we also need to consider the shadows cast on the BTF-mapped surface. A problem with existing rendering techniques is that the mesostructure has no effect near shadow boundaries. Figure 8 (a) shows an example of shadow testing with existing BTF rendering algorithms. The two reference surface points p_1 and p_2 have the same viewing and lighting directions v and l. Without taking the mesostructure into consideration, we would mistakenly regard both p_1 and p_2 as lit.

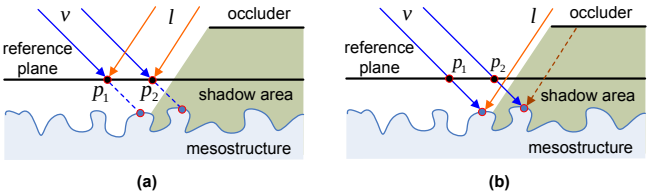

Fig. 8 (a) Without the MDF, both p_1 and p_2 are mistakenly considered lit as indicated by the red arrows. (b) With the MDF, p_1 is lit whereas p_2 is in shadow.

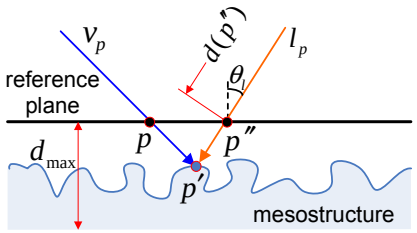

Fig. 9 Shadow computation geometry.

To generate realistic shadows, we need to combine the effects of meso-scale geometry with shadow maps casted by macro-scale objects, which can be easily done with MDF. For simplicity, we assume that we have a point light source casting hard shadows.

As illustrated in Figure 9, for each surface point p with texture coordinate (x,y), we find the intersection point p' of the viewing ray and the mesostructure surface $p' = p + s\ D(x,y,v_p)\ v_p$, where s is the texture scale at p, and v_p is the viewing direction defined in the local coordinate frame. Using the shadow map, we test whether p' is in the macro-scale shadow. Since the self-shadowing effect of mesostructure has been included in BTF, we generate the shadow map for a BTF-mapped surface blocker such that the depth of each point is measured from the "base" of the mesostructure. As shown in Figure 9, at point p'' , the shadow map depth is $d(p'') + \frac{d_{max}}{\cos\theta_l}$, where $d(p'')$ is the depth from the light source calculated using the macro-scale geometry and d_{max} is the maximum depth of the mesostructure.

This algorithm can be easily integrated with the visibility computation algorithm described in the pseudo code. Specifically, to generate the shadow map, we write the $z(p_f) + \frac{d_{max}}{\cos\theta_l}$ into the shadow buffer, where θ_l is the light direction computed in the local coordinate frame at p_f. To render the final result, we compute the mesostructure point with depth is $z(p_f) + d(p_f)$ for p_f and test it against the shadow map. Depending on whether a pixel is in shadow, we shade point p with either an ambient term $L_a(x,y,v_p)$ or the BTF value $T(x,y,v_p,l_p)$, where $L_a(x,y,v_p)$ is obtained by integrating the BTF value for all lighting directions $l = (\theta_l, \phi_l)$ on the hemisphere Ω,

$$L_a(p,v_p) = \int_\Omega T(x,y,v_p,l)\ d\omega_l.$$

5 Experimental Results

Implementation Details: We have implemented our new rendering algorithm on a PC with PIV 2.8GHZ CPU and Geforce 6800GT graphics card. In our implementation, we first render the shadow map for the lighting direction. Then we render the final result with the BTF. To fetch the BTF and MDF data in rendering, we first compute the texture scale s, viewing direction v, lighting direction l, texture coordinate (x,y) and the texture coordinate frame for each vertex. After rasterization, these vertex attributes are interpolated to pixels.

To render the BTF with graphics hardware, we decompose the 6D BTF into several lower dimensional maps using singular value decomposition (SVD) [13]. Specifically, we reorganized the 6D BTF data into a 2D matrix A where the rows are indexed by x ,y, θ_l, ϕ_l and the columns are indexed by viewing angle θ_v,ϕ_v. Applying SVD to A produces a set of 4D Eigen-maps $E_i(x,y,\theta_l,\phi_l)$ and corresponding 2D weight maps $W_i(\theta_v,\phi_v)$. By keeping a small number of Eigen-maps (4 for all BTF data used in this paper), the BTF data are compressed and then can be reconstructed as

$$BTF(x,y,\theta_v,\phi_v,\theta_l,\phi_l) = \sum_i W_i(\theta_v,\phi_v)E_i(x,y,\theta_l,\phi_l)$$

In our current implementation, the 4D Eigen-map is reorganized into a 3D texture for rendering, by the means of packing lighting directions to the third dimension.

To render the 4D MDF data, we directly reorganized the data into 3D texture by packing viewing directions to the third dimension. The tri-linear interpolation is used for texture filtering.

Experimental Results: Table 1 lists all MDF and compressed BTF data used in this paper, where the peanut and the weave data are captured from real materials with our BTF-MDF capturing device. The other data are rendered from synthetic mesostructures by ray-tracing. For BTFs, we store each captured 24-bit RGB image. MDF depth data are quantized to 8 bits per pixel and stored in MDF for rendering. Note that the 4D MDF data is small enough, which can be easily loaded in graphics hardware without compression.

Figure 12 demonstrates a scene rendered with BTFs and MDFs measured from real materials. Both the coarse weave BTF/MDF data on the tori and the peanuts BTF/MDF data on the floor are captured from real materials using our device. Two peeling passes were used to render mesostructure silhouettes and mesostructure shadow silhouettes respectively. Note the mesostructure silhouette of the tori, the detailed shadow boundary on the floor and other visual effects rendered by both BTF and MDF. Also the detailed shadow boundaries cast on the floor is consistently rendered with the implicit geometry rendered with the BTF on the floor.

Figure 13 demonstrates a scene decorated with synthetic grass and brick BTF/MDFs, where three peeling passes are used in rendering. The mesostructure silhouette on the torus knot is convincingly rendered. Also observe the mesostructure shadow silhouette on the floor. Another rendering result of BTF/MDF data is exhibited in Figure 14, where two peeling passes are used to generate the mesostructure silhouette effects.

Sample	Image Res.	Light&View Resolution	BTF Data (MB)	MDF Data (MB)
bluepipe	64×64	$12\times5\times12\times5$	1.3	0.9
brick	64×64	$12\times5\times12\times5$	1.3	0.9
peanut	128×128	$12\times8\times12\times8$	7.5	6.3
weave	128×128	$12\times8\times12\times8$	7.5	6.3
grass	64×64	$12\times8\times12\times8$	5.3	3.9
block	64×64	$12\times5\times12\times5$	1.3	0.9

Table 1 The sampling resolution and size of the BTF/MDF data

Scene	Triangle Number	BTF Only	With Object Silhouette	With Object & Shadow Silhouette
Fig. 1	11,952	186	75	47
Fig. 12	3,200	110	42	27
Fig. 13	11,952	115	30	18
Fig. 14(a)	4,276	186	75	49
Fig. 14(b)	4,276	110	40	25
Fig. 14(c)	4,276	115	85	44

Table 2 The FPS of BTF/MDF rendering for different scenes.

Table. 2 lists the rendering performance of our algorithm for different scenes shown in the paper, where the output image resolution is 800×600.

Limitations: One issue is that the BTF is measured under spatially uniform lighting. Introducing macro-scale blockers causes the illumination to vary spatially. This has a subtle and complicated influence on mesostructure interreflections which our method doesn't account for. Qualitatively, our method generates overly sharp shadows which would be softened if interreflections were simulated accurately. Note that this is a problem for any method based on BTFs. Another issue is that the BTF and MDF data are measured from a plane and then mapped onto curved surfaces. Our peeling-based, depth interval test is only an approximation of the true deformation of the meso-structure geometry onto a curved surface. As shown in figure 11, the method we propose yields consistent results at different surface curvatures. Even in the extreme situation, the bottom-right torus mapped in the tile-2 scale, the meso-structure and silhouette boundary appear matched with each other and consistent as the curvature varies.

Figure 10 compares the rendering results of a cylinder mapped with mesostructure rendered with BTF, BTF/MDF and displaced geometry, respectively. The displaced geometry mapped on the cylinder is rendered by ray tracing. Note that our approach provides convincing visual effects that are consistent with the appearance rendered with the surface BTF. In particular, the shadow generated by our algorithm is very similar to the shadow generated by detailed geometry. The silhouette generated by our method also provides a good approximation of the ground truth.

6 Conclusion

We have presented an algorithm that augments BTF rendering to include previously neglected, detailed silhouette effects using the MDF. The MDF can be measured from real materials without undue additional work in BTF acquisition. Experimental results demonstrate that our new algorithm can render realistic mesostructure silhouettes and shadows, including shadows on BTF-mapped surfaces, in real time and at low storage cost. In future work we are interested in automatically obtaining geometry information solely from the BTF.

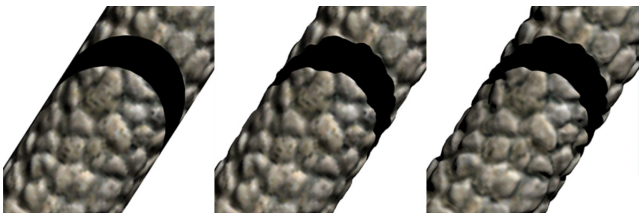

Fig. 10 Comparison of results rendered (a) by BTF only (b) by BTF and MDF (c) ground truth.

Acknowledgements We are grateful to the anonymous reviewers, whose comments were very valuable in refining our paper. Many thanks to Yasuyuki Matsushita for helping us setup capture devices.

References

1. Blinn, J.F.: Simulation of wrinkled surfaces. Computer Graphics (SIGGRAPH '78 Proceedings) **12**(3), 286–292 (1978)
2. Bouguet, J.Y., Perona, P.: 3d photography on your desk. In: IEEE International Conference on Computer Vision (ICCV98), pp. 43–50 (1998)
3. Cook, R.L.: Shade trees. Computer Graphics (SIGGRAPH '84 Proceedings) **18**(3), 223–231 (1984)
4. Curless, B., Levoy, M.: Better optical triangulation through space-time analysis. In: IEEE International Conference on Computer Vision (ICCV95), pp. 987–994 (1995)
5. Dana, K.J.: Brdf/btf measurement device. In: Proceedings of eighth IEEE international conference on computer vision (ICCV), vol. 2, pp. 460–466 (2001)
6. Dana, K.J., van Ginneken, B., Nayar, S.K., Koenderink, J.J.: Reflectance and texture of real-world surfaces. ACM Transactions on Graphics **18**(1), 1–34 (1999)
7. Diefenbach, P.J.: Pipeline Rendering: Interaction and Realism through Hardware-Based Multi-Pass Rendering. Ph.D. thesis (1996)
8. Furukawa, R., Kawasaki, H., Ikeuchi, K., Sakauchi, M.: Appearance based object modeling using texture database: Acquisition,

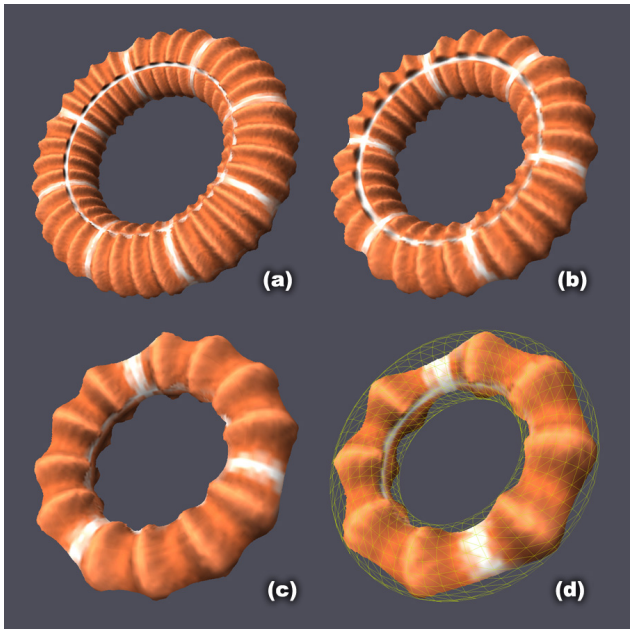

Fig. 11 Rendering results of the same torus mapped with brick BTF/MDF data in different texture scales. (a) with 8×3 tiles (b) with 6×2 tiles (c) with 3×1 tiles (d) with 2×1 tiles. The base triangle mesh is showed as the yellow wireframes in (d).

compression and rendering. In: Eurographics Workshop on Rendering, pp. 257–266 (2002)
9. Han, J.Y., Perlin, K.: Measuring bidirectional texture reflectance with a kaleidoscope. ACM Transactions on Graphics **22**(3), 741–748 (2003)
10. Heidrich, W., Daubert, K., Kautz, J., Seidel, H.P.: Illuminating micro geometry based on precomputed visibility. In: Computer Graphics (Proceedings of SIGGRAPH 2000), pp. 455–464 (2000)
11. Hirche, J., Ehlert, A., Guthe, S., Doggett, M.: Hardware accelerated per-pixel displacement mapping. In: GI '04: Proceedings of the 2004 conference on Graphics interface, pp. 153–158. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (2004)
12. Kaneko, T., Takahei, T., Inami, M., Kawakami, N., Yanagida, Y., Maeda, T., Tachi, S.: Detailed shape representation with parallax mapping. In: ICAT2001(11th International Conference on Artificial Reality and Tele-Existence), pp. 205–208 (2001)
13. Kautz, J., McCool, M.D.: Interactive rendering with arbitrary BRDFs using separable approximations. In: D. Lischinski, G.W. Larson (eds.) Rendering Techniques '99, Eurographics, pp. 247–260. Springer-Verlag Wien New York (1999)
14. Koenderink, J.J., Doorn, A.J.V.: Illuminance texture due to surface mesostructure. Journal of the Optical Society of America **13**(3), 452–463 (1996)
15. Liu, X., Hu, Y., Zhang, J., Tong, X., Guo, B., Shum, H.Y.: Synthesis and rendering of bidirectional texture functions on arbitrary surfaces. IEEE Transactions on Visualization and Computer Graphics **10(3)**, 278–289 (2004)
16. Liu, X., Yu, Y., Shum, H.Y.: Synthesizing bidirectional texture functions for real-world surfaces. Proceedings of SIGGRAPH 2001 pp. 97–106 (2001)
17. Malzbender, T., Gelb, D., Wolters, H.: Polynomial texture maps. Proceedings of SIGGRAPH 2001 pp. 519–528 (2001)
18. Mammen, A.: Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. IEEE Computer Graphics and Applications **9**(4), 43–55 (1989)
19. Max, N.: Horizon mapping: Shadows for bumped mapped surfaces. The Visual Computer pp. 109–117 (1988)
20. Mueller, G., Meseth, J., Klein, R.: Compression and real-time rendering of measured BTFs using local PCA. In: Proceedings of Vision, Modeling and Visualisation (2003)
21. Policarpo, F., Oliveira, M.M., Comba, J.L.D.: Real-time relief mapping on arbitrary polygonal surfaces. In: ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games (I3D 2005) (2005)
22. Rushmeier, H., Balmelli, L., Bernardini, F.: Horizon map capture. In: Proceedings of Eurographics 01 (2001)
23. Sattler, M., Sarlette, R., Klein, R.: Efficient and realistic visualization of cloth. In: Eurographics Symposium on Rendering, pp. 167–178 (2003)
24. Sloan, P.P., Cohen, M.F.: Interactive horizon mapping. In: Eurographics Workshop on Rendering, pp. 281–286 (2000)
25. Sloan, P.P., Liu, X., Shum, H.Y., Snyder, J.: Bi-scale radiance transfer. ACM Transactions on Graphics **22**(3), 370–375 (2003)
26. Suykens, F., vom Berge, K., Lagae, A., Dutré, P.: Interactive rendering with bidirectional texture functions. In: Proceedings of Eurographics 03 (2003)
27. Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., Shum, H.Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. ACM Transactions on Graphics **21**(3), 665–672 (2002)
28. Vasilescu, M.A.O., Terzopoulos, D.: TensorTextures: multilinear image-based rendering. ACM Transactions on Graphics **23**(3), 336–342 (2004)
29. Wang, L., Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., Shum, H.Y.: View-dependent displacement mapping. ACM Transactions on Graphics **22**(3), 334–339 (2003)
30. Wang, X., Tong, X., Lin, S., Hu, S., Guo, B., Shum, H.Y.: Generalized displacement maps. In: Eurographics Symposium on Rendering 2004 (2004)
31. Yamauchi, Y., Sekine, M., Yanagawa, S.: Bidirectional texture mapping for realistic cloth rendering. SIGGRAPH 2003 Sketches and Applications (2003)
32. Zhang, Z.: Flexible camera calibration by viewing a plane from unknown orientations. In: IEEE International Conference on Computer Vision (ICCV99), pp. 666–673 (1999)

Jiaping Wang Jiaping Wang is a first year Ph.D. student of Institute of Computing Technology, Chinese Academy of Sciences in Graduate School of the Chinese Academy of Sciences. His advisor is Heung-Yeung Shum. Before that, he received B.S. Degree from Ningbo university in 2002. He has been taking internship in Microsoft Research Asia (MSRA) since 2003.

His research interests include appearance modelling and capturing, hardware accelerated rendering, high-dimensional texture analysis and compression. He is also interested in early vision, perception, and machine learning.

Xin Tong Dr. Tong is a researcher/project lead of Internet graphics group, Microsoft Research Asia. After received his Ph.D. Degree in Computer Graphics from Tsinghua University, Beijing in July 1999, He joined Microsoft Research China as an associate researcher. Before that, He received his B.S. Degree and Master Degree in Computer Science from Zhejiang University in 1993 and 1996 respectively.

His research interests includes appearance modelling and rendering, image-Based rendering, texture synthesis and natural phenomena simulation.

John Snyder Dr. Snyder is a Senior Reseacher at Microsoft Research. He received a B.S. from Clarkson University in 1984, a Ph.D. from the California Institute of Technology in 1991, and has been at Microsoft Research since 1994. His research interests include geometry representation and processing and real-time algorithms for global illumination.

Yanyun Chen Dr. Chen is an associate researcher of Microsoft Research Asia. His current research interests include photorealistic and non-photorealistic rendering and image-based rendering. Dr. Chen received his Ph.D. from the Institute of Software, Chinese Academy of Sciences in 2000, and his M.S. and B.S. from the Chinese University of Mining and Technology in 1996 and 1993, respectively.

Baining Guo Dr. Guo is the research manager of the internet graphics group at Microsoft Research Asia. Before joining Microsoft, Baining was a senior staff researcher in Microcomputer Research Labs at Intel Corporation in Santa Clara, California, where he worked on graphics architecture. Baining received his Ph.D. and M.S. from Cornell University and his B.S. from Beijing University. Baining is an associate editor of IEEE Transactions on Visualization and Computer Graphics. He holds over 30 granted and pending US patents.

Heung-Yeung Shum Dr. Shum is currently the Managing Director of Microsoft Research Asia (MSRA). Dr. Shum joined Microsoft Research in 1996, after receiving a Ph.D. in Robotics from the School of Computer Science at Carnegie Mellon University. He has authored and co-authored over 100 papers in computer vision, computer graphics and robotics, and received more than 20 US patents. He is on the editorial boards for IEEE Transactions on Circuit System Video Technology (CSVT), IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), International Journal of Computer Vision, and Graphical Models.

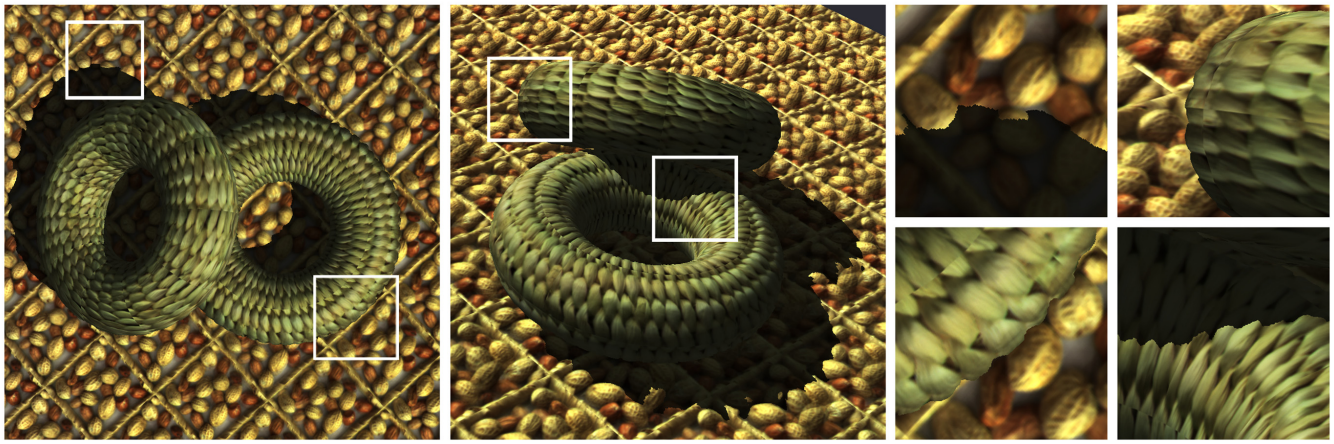

Fig. 12 Rendering result of BTF-mapped surfaces with real-world BTF/MDF samples.

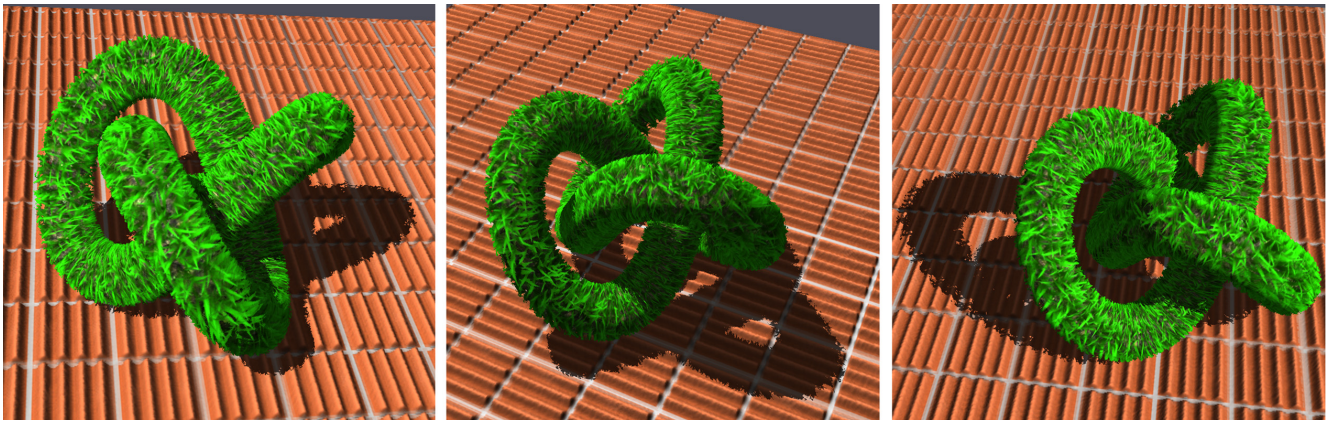

Fig. 13 BTF mapped surface with synthetic BTF and MDF data.

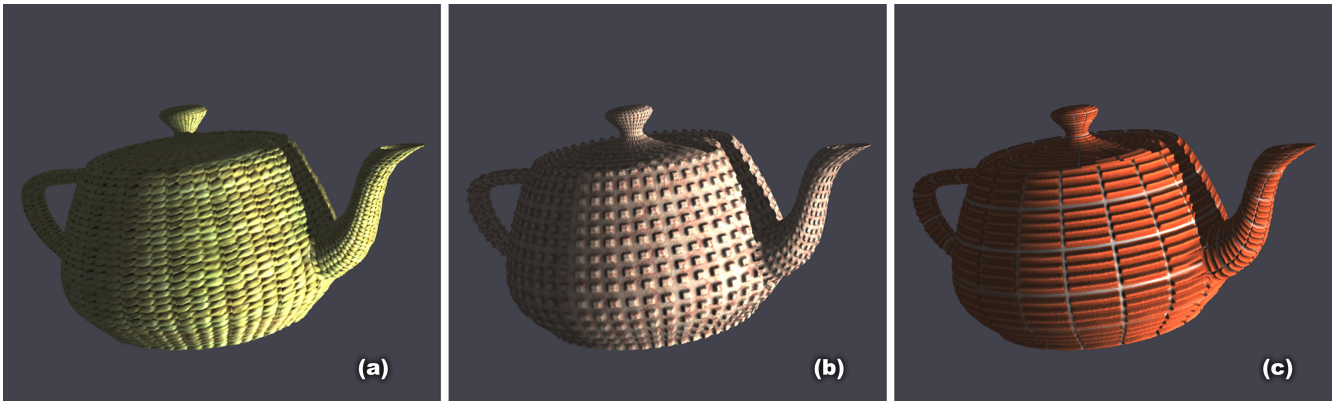

Fig. 14 Three examples of BTF-mapped surfaces with shadow and silhouette effects. (a) with weave BTF/MDF (b) with block BTF/MDF (c) with brick BTF/MDF.