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

Ying Song · Yanyun Chen · Xin Tong · Stephen Lin · Jiaoying Shi · Baining Guo · Heung-Yeung Shum

Shell Radiance Texture Functions

Abstract The appearance of an inhomogeneous translucent material depends substantially on its volumetric variations and their effects upon subsurface scattering. For efficient rendering that accounts for both surface mesostructures and volumetric variations of such materials, shell texture functions have precomputed irradiance within a volume with respect to incoming illumination, but even with this irradiance data a fair amount of run-time computation is still required. Rather than precompute volume irradiance, we introduce the shell radiance texture function (SRTF) which relates incoming illumination more directly to outgoing surface radiance by representing a set of subsurface transport components from which surface radiance can be calculated without ray marching or run-time evaluation of dipole diffusion. Using this precomputed SRTF information, inhomogeneous objects can be rendered in real-time with distant local lighting or global lighting.

Keywords subsurface scattering · mesostructure · texture mapping · real time rendering

1 Introduction

The realism of rendered images can be significantly enhanced by incorporating visual effects such as light propagation within translucent objects, and shadowing, masking, and interreflection by surface mesostructures. To render the effects of translucency, full physical simulations of subsurface scattering by Monte Carlo methods provide high quality results but at a considerable expense in computation [1,11,14], especially for illumination and viewing directions that vary from frame to frame. For greater efficiency, Jensen et al. [12] propose the dipole approximation based on diffusion theory [23] for optically dense homogeneous materials in which multiple scattering dominates. Although this method reduces rendering times from hours down to seconds, it cannot handle the vast range of common materials that are inhomogeneous.

To render the appearance details of surface mesostructures, many texture functions have been introduced for handling the interactions of mesostructures with illumination. These methods, which include height field displacements [2], volumetric textures [18], and bidirectional texture functions (BTFs) [5], mainly focus on surface geometry and cannot fully capture object translucency effects, which is especially evident for backlighting.

The shell texture function (STF) [4] has recently been introduced to render both object mesostructures and translucency. It proposes a two-layer model consisting of an inhomogeneous outer shell and a homogeneous inner core. The shell is synthesized from a base volume with spatially-variant subsurface scattering parameters specified by the user. Each voxel of the shell stores irradiance precomputed by photon mapping, while the inner core is evaluated by the dipole diffusion approximation [12]. In rendering, ray marching is applied from the top to the bottom voxels of the shell, and the final radiance is integrated along the ray. STFs are hard to implement in real-time for two reasons. First, it uses ray marching to gather the radiance along a viewing ray through the volume, which is a time-consuming process. Second, it computes the dipole approximation at run time, which also slows rendering.

In this paper, we employ the same object model as in STF, but to enable real-time rendering we precompute the shell radiance texture function (SRTF), which consists of 6D subsurface transport components that are

Ying Song, Jiaoying Shi
Zhejiang University
Tel.: +086-571-88206681
Fax: +086-571-88206680
E-mail: songying, jyshi@cad.zju.edu.cn

Yanyun Chen, Xin Tong, Stephen Lin, Baining Guo, Heung-Yeung Shum
Microsoft Research Asia
Tel.: +086-10-62617711
Fax: +86-10-88097306
E-mail: yachen, xtong, stevelin, bainguo, hshum @microsoft.com

This work was done while Ying Song was visiting Microsoft Research Asia from the Zhejiang University.

directly related to surface radiance. Using the layered model, we efficiently divide radiance into shell and core contributions. While the low-frequency core radiance contribution can be precomputed and stored on each vertex, the detailed radiance contribution from the inhomogeneous shell can be evaluated more finely by per pixel computation. By offline computation of radiance-based components on the surface instead of irradiance values within the volume, ray marching is avoided at run time. To deal with the higher dimensionality of 6D radiance data in comparison to 5D irradiance data, we propose compression methods for local illumination and global illumination that greatly reduce the memory usage. For more efficient evaluation of the dipole diffusion approximation, we present a precomputation scheme to minimize run-time processing. With the precomputed SRTF and precomputed dipole diffusion, objects with surface mesostructures and inhomogeneous translucency can be rendered in real time.

2 Related Work

Mesostructures contribute significantly to surface appearance. While image-based representations such as BTFs [5] and surface light fields [24] have been proposed for mesostructure modeling, they provide a representation of subsurface scattering that is accurate only for the specific object geometries under which they are acquired. A good survey on real-time BTF rendering can be found in [15]. Traditional rendering techniques for mesostructures such as view-dependent displacement maps [26] and generalized displacement maps [25] have also been presented, but they are designed to handle only opaque materials.

Efficient methods for rendering translucent materials have generally been based on the dipole diffusion approximation presented by Jensen et al. [12]. Jensen and Buhler [10] propose a hierarchical approach where subsurface scattering can be evaluated in a few seconds, which is far more efficient than Monte-Carlo methods but still not real-time.

Several subsequent techniques achieve interactive or real-time performance based on Jensen's dipole approximation. Lensch et al. [13] introduced a method based on precomputation to render translucent materials, and has been used to render inhomogeneous objects captured from the real world [7]. Their preprocessing is divided into a local and a global part. The local part computes pixel-to-pixel scattering throughput factors accounting for subsurface scattering from the immediate neighborhood, while the global part computes vertex-to-vertex throughput factors accounting for subsurface scattering over a greater distance. In rendering, both the local and global parts are convolved with the illumination map over the entire surface. Because of substantial computation on the CPU, only interactive frame rates can be achieved.

In [3], Carr et al. evaluate subsurface scattering on the GPU with a hierarchical radiosity approach. The scattering throughput factor which represents subsurface scattering between two surface patches is modeled as a link between two areas in the geometry atlas. These links are used to evaluate the scattered radiance in the rendering process. In rendering, an illumination map is first created, storing the irradiance on each pixel of the geometry atlas. The illumination map is then integrated with the precomputed links to obtain the scattered radiance map. Finally the scattered radiance map is scaled by the Fresnel term to obtain the final result. The approach is fully implemented on hardware, and the links are stored as textures. Only a few links are employed for real-time evaluation of subsurface scattering due to hardware limitations, which results in less accurate local variation. Adding more links, however, would lead to slower rendering performance.

Hao et al. [8,9] render translucent materials by precomputing an integral of the BSSRDF [12] with a local illumination model. The precomputation is performed on each vertex, and is integrated over the local neighborhood of the vertex. Only the local subsurface scattering contribution is accounted for. To extend to inhomogeneous materials, a dense mesh must be utilized to preserve local variation, which would slow down performance significantly.

Sloan et al. [21] use precomputed radiance transfer (PRT) to precompute subsurface scattering. Since PRT applies per-vertex radiance transfer, preserving local appearance variations requires dense vertices, which reduces rendering speed. They later present bi-scale precomputed radiance transfer to maintain local appearance variations [22], but this framework is designed only for opaque objects.

Translucent shadow maps (TSMs) [6] render translucent objects with local illumination based on standard shadow maps. In rendering, subsurface scattering is computed by filtering the shadow map neighborhood. TSM can only render homogeneous materials because they use the dipole approximation, and this approach is difficult to extend to global illumination.

Premože et al. [19] introduce an interactive rendering method for translucent media using path integration. This approach is only valid for sparse inhomogeneous media with smoothly varying scattering coefficients. In contrast, our method mainly focuses on optically dense inhomogeneous materials which may have sharp changes in scattering properties.

3 SRTF Model

As illustrated in Figure 1, the object model of the SRTF is the same as that of the STF, where an inhomogeneous translucent object is divided into an inhomogeneous shell and a homogeneous core. The shell is formed by texture

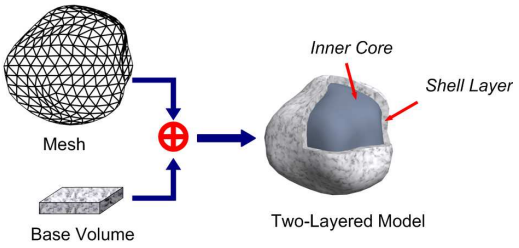

Fig. 1 The two-layered object model.

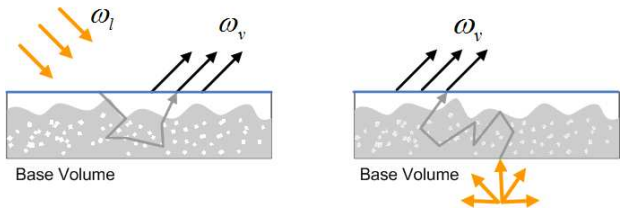

Fig. 2 The two SRTF components, f_{shell} (left) and f_{core} (right).

synthesis of a volumetric material sample called the base volume, as described in [4]. In contrast to the STF where irradiance values are precomputed within the base volume and then used to render the surface radiance by ray marching at run time, we precompute the SRTF from the base volume, which is then used to directly evaluate the surface radiance at run time. Light transport through the core is modelled by the dipole diffusion approximation, whose contribution is computed and stored on the mesh before the rendering.

3.1 SRTF Construction

The SRTF is precomputed from the base volume with the following material properties for each voxel: extinction coefficient σ_t, albedo α, and phase function $p(\vec{\omega}, \vec{\omega}')$. The scattering coefficient is related to the extinction coefficient and the albedo by $\sigma_s = \alpha\sigma_t$. The transmittance between x and x' is defined as $\tau(x, x') = e^{-\int_x^{x'} \sigma_t(u)du}$. As described in [4], the material parameters within a base volume can be acquired by computed tomography (CT) or designed with 3D modeling tools.

Different from the STF which is defined in the base volume, the SRTF is defined on the reference plane that lies on the top of the base volume. As an image-based representation, SRTF directly records the outgoing radiance from the base volume for all viewing and lighting directions. At rendering time, the precomputed SRTF is directly obtained from the object surface and thus avoids ray marching.

As illustrated in Figure 2, the SRTF is composed of two components, f_{shell} and f_{core}. For each incoming light ray along direction $\omega_l = (\theta_l, \phi_l)$ that samples the upper hemisphere, the outgoing radiance towards direction $\omega_v = (\theta_v, \phi_v)$ is represented by $f_{shell}(x, y, \theta_v, \phi_v, \theta_l, \phi_l)$, where (x, y) is a sampling point on the reference plane. f_{shell} records both scattering effects within the base volume and inter-reflection, masking and shadowing effects caused by surface mesostructure. Likewise, f_{core} models radiance in the outgoing direction due to light arriving from the lower hemisphere. Since in rendering the backlighting from the lower hemisphere comes from the diffuse radiance of homogeneous inner core, f_{core} is related only to the viewing direction and is independent of the lighting direction.

As illustrated in Figure 3, for sampled illumination directions, we compute f_{shell} in two steps using the material properties of the base volume. In the first step, we employ photon mapping proposed in [11] to compute the in-scattered radiance within the volume and the irradiance on the mesostructure surfaces. From the mapped photons, we then evaluate the surface radiance for each viewing direction by ray tracing. f_{core} is modelled in the same way by using diffuse light from the entire lower hemisphere.

3.2 Layered Object Modeling

Given the base volume and target mesh, we synthesize or map the base volume onto the target mesh to form the two-layered model, as described in [4]. In synthesis, we align the top voxels of the base volume with the target mesh surfaces. After that, each surface point on the target mesh is assigned a texture coordinate and corresponds to a point on the reference plane. As a result, the SRTF values associated with the reference plane are assigned to the surface.

3.3 Core Radiance Precomputation

Based on the two-layer object model, we divide surface radiance into two parts: from light scattered only within the shell, and from light scattered from the core, as illustrated in Figure 4. Using f_{shell}, surface radiance from shell scattering can be directly evaluated at run time. To compute light scattered from the inner core, we combine the precomputed core radiance L_c with f_{core}, which

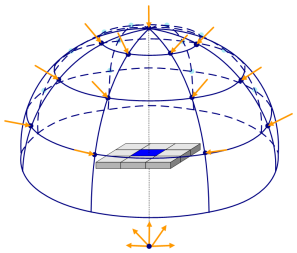

Fig. 3 Sampling the SRTF from the base volume. To avoid boundary effects in sampling, the base volume is surrounded by eight other identical volumes as done in [4].

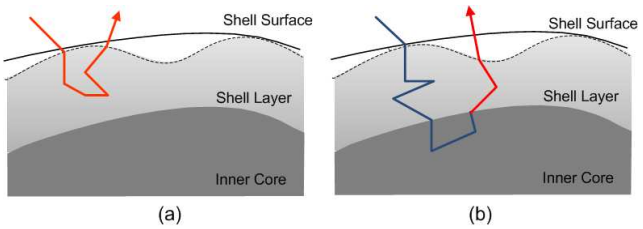

Fig. 4 Light scattering in the layered model. (a) Light scattered within the shell. (b) Light scattered through the core. The blue part is precomputed by core radiance, while the red part is evaluated with f_{core}.

models the scattering of light from the inner core through the outer shell.

To precompute the core radiance for incident light on the surface, we must first account for the effects of light transmission through the shell to compute the core irradiance. To accelerate the core irradiance computation, we precompute $f_{irra}(x, y, \theta_l, \phi_l)$, which records for each sampled light direction from $\omega_l = (\theta_l, \phi_l)$ the multiple scattering irradiance arriving at each voxel (x, y) that is on the bottom level of the base volume. This irradiance value is sampled from the base volume by photon mapping and then used in the following core radiance precomputation.

Since multiple scattering is dominant in the homogeneous inner core, we then use the dipole diffusion approximation R_d [12] to compute the core radiance L_c from the core irradiance. In our computation, we ignore the thickness of the shell layer and directly compute the core radiance over the mesh surface. For efficient determination of the dipole approximation, we employ a precomputation scheme with respect to directional lighting. Specifically, for incoming illumination from direction ω_l, the core radiance at a surface point x_o can be computed by:

$$L_c(x_o, \omega_l) \approx \frac{1}{\pi} \int_A R_d \|x_i - x_o\| f_{irra}(T(x_i), \omega_l') V(x_i, \omega_l) L(\omega_l) d(A(x_i))$$

where R_d is evaluated as done in [12]. x_i is a point on the surface, and $A(x_i)$ is a small area around x_i. $V(x_i, \omega_l)$ is the visibility of light at x_i and ω_l' is the light direction computed in the local coordinate frame of x_i. $T(x_i)$ is the texture coordinate of x_i. We uniformly sample over the entire sphere of incoming directions and precompute this integral for each sampled light direction. Due to the homogeneity of the inner core, the resulting radiance varies slowly over the surface. Therefore we precompute the core radiance on each mesh vertex. The set of precomputed integrals on each vertex is then compressed using spherical harmonics (SH). In rendering, L_c is reconstructed from a dot product with spherical harmonic lighting coefficients. Since L_c is a low-frequency diffuse term, this scheme works even for a local illumination model, as done in [9]. But while [9] represents scattering in a local volume, our technique accounts for scattering throughout the entire object. In our implementation, we use a 4th-order spherical harmonic compression of the core radiance. As shown in Figure 5, it is visually indistinguishable from the original one.

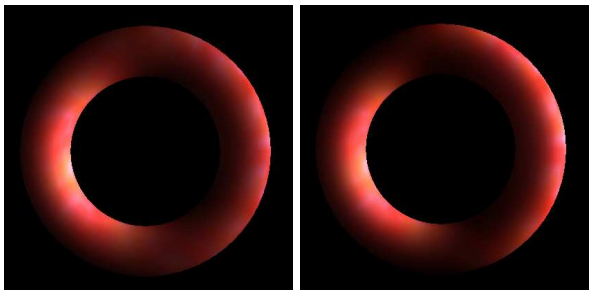

Fig. 5 Comparison between the core radiance recovered from 4th-order SH-compressed coefficients for directional lighting (left) and core radiance directly computed from the input irradiance (right).

4 Rendering with SRTF

Rendering the layered model with SRTF is straightforward. From the SRTF and precomputed core radiance L_c, surface radiance can be rapidly computed at run time by combining the radiance contributions from the shell and core according to

$$L(x, \omega_v) = \int_\Omega (f_{shell}(T(x), \omega_v, \omega_l) + f_{core}(T(x), \omega_v) L_c(x, \omega_l)) L(\omega_l) d\omega_l$$

where x is a surface point and $T(x)$ is the texture coordinate of x. ω_v is the viewing direction at x.

To render the layered model with graphics hardware, we decompose and compress the 6D f_{shell} into several lower dimensional texture functions. For the relatively small 4D f_{core} data, we pack the two dimensions of the lighting direction into one dimension and reorganize it as a 3D texture, which is used for rendering directly.

4.1 Rendering with Local Illumination

For local illumination, we assume the lighting to be distant and directional. To fit f_{shell} into the graphics memory, we compress f_{shell} data based on Principal Component Analysis(PCA)[16,17,20] using Singular Value Decomposition (SVD). Specifically, we pack each 4D subset f_{shell} sampled under the same viewing direction as a 2D matrix A, in which each row is the image sampled from one lighting direction. Applying SVD to A generates a set of 2D Eigen-maps $E_l^v(x, y)$ and 2D weight

maps $W_l^v(\theta_l, \phi_l)$. By keeping a small number of Eigenmaps (8 in our current implementation), the f_{shell} data are decomposed and compressed as a set of 2D maps. In rendering, f_{shell} can be reconstructed by graphics hardware as

$$f_{shell}(x, y, \theta_v, \phi_v, \theta_l, \phi_v) = \sum_i W_i^v(\theta_l, \phi_l) E_i^v(x, y).$$

To render the layered model under local illumination, we first compute the viewing direction v, lighting direction l, texture coordinate $T(x)$ and the local coordinate frame for each vertex. The core radiance L_c is also evaluated at each vertex by projecting the directional light onto the SH basis and computing its dot product with the precomputed core radiance SH coefficient vector stored on the vertex. After rasterization, these vertex attributes are interpolated to pixels. In the pixel shader, the interpolated vertex attributes are used to fetch and evaluate the f_{core} and f_{shell} values for each pixel. After scaling f_{core} by L_c in each pixel, we combine the radiance contributions from shell and core to generate the final result.

4.2 Rendering with Global Illumination

For global illumination, we employ the PRT framework [22] to render the layered model illuminated with the SH-based environment map. To apply f_{shell} with SH-based environment lighting, we project f_{shell} for each viewing direction and compute the SH coefficients for each viewing direction by

$$f_l^m(x, \omega_v) = \int_\Omega f_{shell}(x, \omega_l, \omega_v) y_l^m(\omega_l) d\omega_l$$

where $y_l^m(\omega_l)$ is the real-valued SH basis. By keeping the low orders of SH coefficients (4 in our current implementation), the f_{shell} data is compressed.

Before rendering, we precompute a radiance transfer matrix M_T in the local frame for each vertex, which accounts for visibility when transforming the environment map illumination to transferred incidence radiance.

In rendering, the High Dynamic Range(HDR) environment map is first projected onto the SH basis to obtain a SH light vector L' by keeping the 4th-order coefficients. We then multiply the M_T with light vector L' to get the incident radiance vector at each vertex. The SH coefficients of the incident radiance vector are assigned to a vertex as vertex attributes and then interpolated to pixels after rasterization. In the pixel shader, we compute the contribution of the shell by a dot product $L_{shell} = f'_{shell} \cdot M_T \cdot L'$ where f'_{shell} is a SH coefficient vector of f_{shell}. The final radiance is computed as $L = L_{shell} + L_c \cdot f_{core}$, where core radiance is evaluated in the same manner as for local illumination.

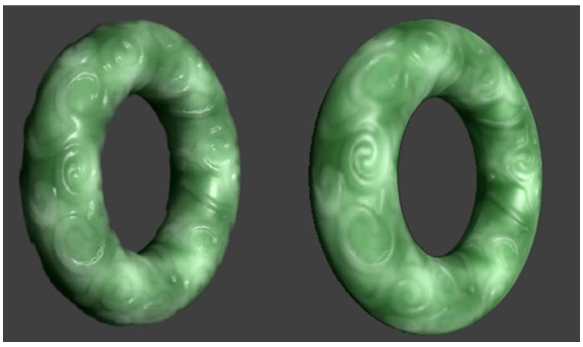

Fig. 6 Comparison between the result rendered by STF (left) at about one frame per minute, and result rendered by real-time SRTF (right).

5 Experimental Results

We tested our algorithm on a 2.4GHZ Pentium IV PC with an ATI Radeon 9800 128MB graphics card. SRTF rendering is implemented using vertex/pixel shader 2.0 with Direct3D.

In Figure 8(a), a torus made of translucent jade-like material is rendered both with local illumination and with global illumination. The radiance contributions from subsurface scattering in the outer shell and the inner core have a clear effect on appearance. Figure 8(b) exhibits rendering results for a bird model. The back-lighting effects caused by subsurface scattering through this inhomogeneous volume are evident. Also note that the masking effects caused by mesostructure on the surface. Figure 8(c) displays rendering results of a bunny model with high frequency surface details and material variations, which are challenging for existing techniques [9, 22].

A rendering comparison between STFs and SRTFs is shown in Figure 6. SRTFs exhibit slight blurring in comparison to STFs because of compression, and silhouettes are not included. However, real-time rates are achieved with SRTF, while rendering with STF requires approximately one minute per frame. With the material in Figure 6, the data storage of the STF is 22MB, and that of the SRTF is 64MB with the same sampling resolution before compression. After compression, the SRTF will be 21MB.

Table 1 lists the rendering performance and SRTF resolution for the three models shown in Figure 8. The rendering performance is tested for a 512×512 output window. The compression ratio of SRTF data is about 3:1 for both local and global illumination. Figure 7 compares the results under different compression rates, using PCA and SH respectively. We can conclude that 8-terms in PCA and 4th-order in SH compression can achieve acceptable results.

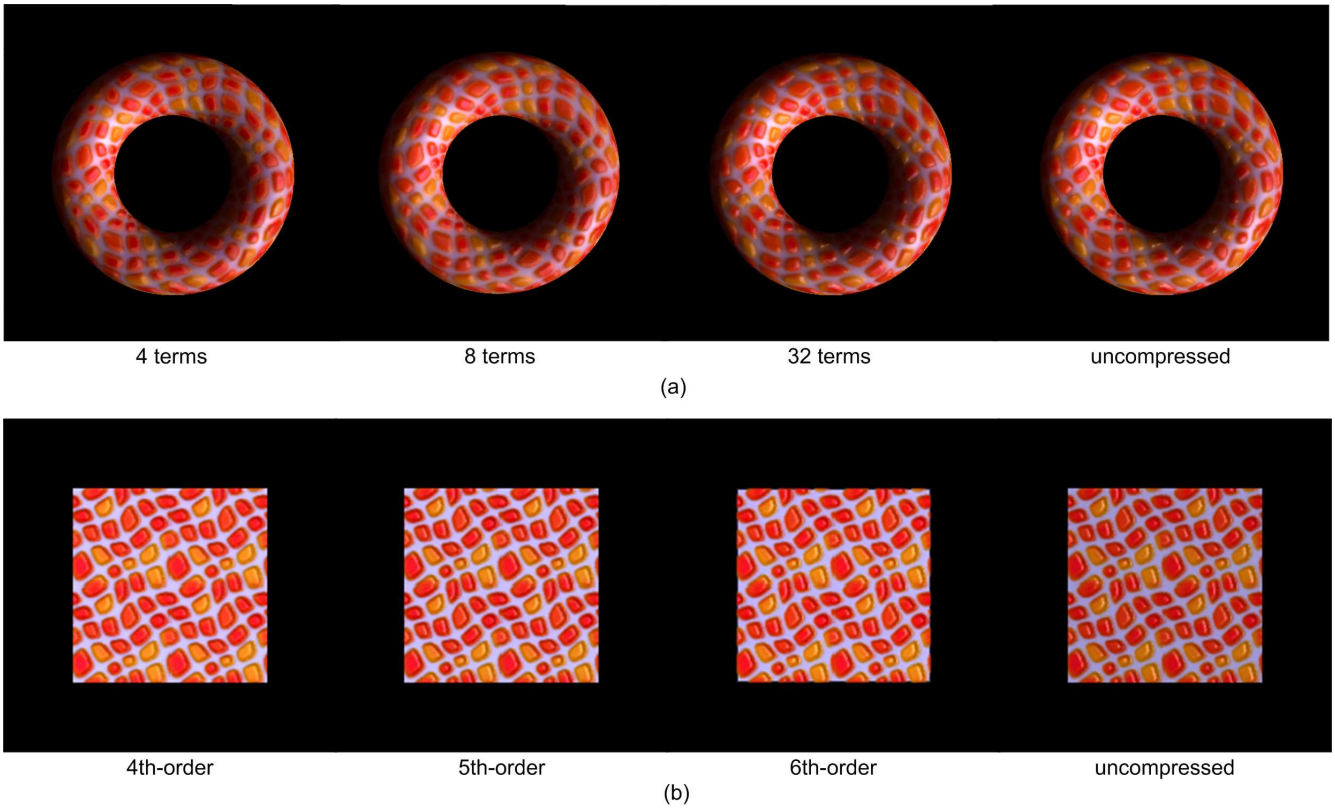

Fig. 7 Comparison between results rendered with SRTF at different compression rates. (a) using PCA in compression, and (b) using SH in compression.

Table 1 Rendering performance of three models with different materials.

Model	Vertices	FPS		SRTF Parameters		
		Local	Global	View Res.	Light Res.	Image Res.
Bird	35k	31	17	8×8	8×8	64×64
Bunny	35k	29	12	12×8	12×8	64×64
Torus	10k	41	36	8×8	8×8	128×128

6 Conclusion

Rendering of inhomogeneous translucent objects with complex surface mesostructures and arbitrary shape has been a challenging problem in computer graphics, and with the SRTF, they can be rendered in real time. The SRTF also has some limitations. SRTF is not suitable for rendering very translucent objects. In addition, the detailed silhouette caused by surface mesostructure is ignored in SRTF rendering. We are investigating how to combine an existing silhouette rendering approach (such as [25]) with SRTF rendering. Another direction of future work is to extend the SRTF to represent and render arbitrary existing objects of non-homogeneous materials, which can presently be rendered only by full participating media simulation now.

Acknowledgements We would like to thank the anonymous reviewers for their valuable comments in refining our paper. Many thanks to Xi Wang for helping make the video, and Jiaping Wang for the helpful discussion on the implementation. Ying Song and Jiaoying Shi were partially supported by National 973 Project No.2002CB312105.

References

1. P. Blasi, B. Le Saec, and C. Schlick. An Importance Driven Monte-Carlo Solution to the Global Illumination Problem. In: Eurographics Workshop on Rendering ,173–183. (1994)
2. R.L Cook: Shade Trees. In: Computer Graphics, 223–331. (1984)
3. N.A. Carr, J.D. Hall, and J.C. Hart: GPU Algorithms for Radiosity and Subsurface Scattering. In:Proc. Graphics Hardware, 51–59. (2003)
4. Y. Chen, X. Tong, J. Wang, S. Lin, B. Guo and H.Y. Shum. : Shell Texture Functions. In: Proc. ACM SIGGRAPH, 343–353. (2004)
5. K.J. Dana, B.V. Ginneken, S.K. Nayar, and J.J. Koenderink. : Reflectance and Texture of Real-World Surfaces. In: ACM Transactions on Graphics. (1999)
6. C. Dachsbacher and M. Stammingerz: Translucent Shadow Maps. In: Rendering Techniques, 197-201. (2003)

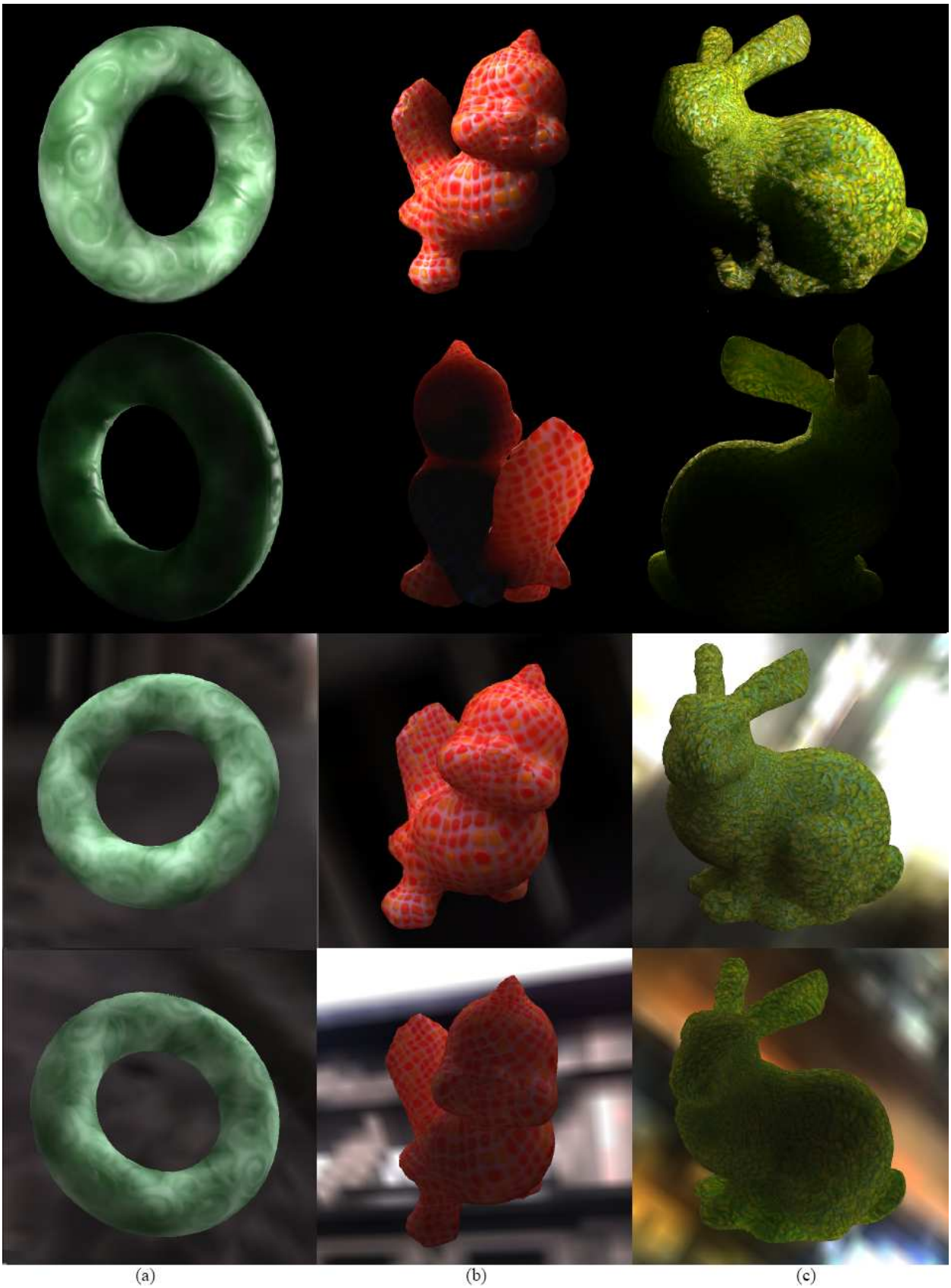

Fig. 8 Rendering results of three models, each in one column. The top two rows illustrate the rendering result in local illumination, while the bottom two rows illustrate the result in global illumination, where uses high-dynamic environment maps.

7. M. Goesele, H.P.A. Lensch, J. Lang, C. Fuchs, H.-P. Seidel.: DISCO - Acquisition of Translucent Objects. In: Proc. ACM SIGGRAPH, 835–844. (2004)
8. X. Hao, T. Baby, and A. Varshney.: Interactive subsurface scattering for translucent meshes. In: Symposium on Interactive 3D Graphics, 75–82. (2003)
9. X. Hao and A. Varshney.: Real-time rendering of translucent meshes. In: ACM Transactions on Graphics, vol 23, 120–142. (2004)
10. H.W. Jensen and J. Buhler: A rapid hierarchical rendering technique for translucent materials. In: ACM Transaction on Graphics 21(3), 576-581. (2002)
11. H.W. Jensen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In: Proc. ACM SIGGRAPH, 311–320. (1998)
12. H.W. Jensen, S.R. Marschner, M. Levoy, and P. Hanrahan.: A practical model for subsurface light transport. In Proc. ACM SIGGRAPH, 511–518. (2001)
13. H.P.A. Lensch, M. Goesele, P. Bekaert, J. Kautz, M.A. Magnor, J. Lang, H.P. Seidel.: Interactive rendering of translucent objects. In: Proc. Pacific Graphics, 214–224. (2002)
14. E.P. Lafortune and Y.D. Willems.: Rendering Participating Media with Bidirectional Path Tracing. In: Rendering Techniques, 91–100. (1996)
15. G. Müller, J. Meseth, M. Sattler, R. Sarlette, R. Klein.: Acquisition, Synthesis and Rendering of Bidirectional Texture Functions. In: Computer Graphics forum, Vol. 24, No. 1, 83-109. (2005)
16. G. Müller, J. Meseth, R. Klein.: Fast Environmental Lighting for Local-PCA Encoded BTFs. In: proceedings of Computer Graphics International, 198-205. (2004)
17. W-C. Ma, S-H. Chao, Y-T. Tseng, Y-Y.Chuang, C-F Chang, B-Y. Chen and M. Ouhyoung.: Level-of-detail Representation of Bidirectional Texture Functions for Real-time Rendering. In: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, 187-194.(2005)
18. F. Neyret.: Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures. In: IEEE Trans. Visual Computer Graphics, 4(1), 55-70. (1998)
19. S. Premože, M. Ashikhmin, J. Tessendorf, R. Ramamoorthi, and S. Nayar.: Practical rendering of multiple scattering effects in participating media. In: Eurographics Symposium on Rendering. (2004)
20. M. Sattler, R. Sarlette, and R. Klein. In:Effcient and realistic visualization of cloth. In: Proceedings of Eurographics Symposium on Rendering, 167-177.(2003)
21. P.P. Sloan, J. Hall, J. Hart, and J. Snyder.: Clustered principal components for precomputed radiance transfer. In: ACM Transactions on Graphics, vol 22-3, 382–391. (2003)
22. P.P. Sloan, X. Liu, H.Y. Shum, and J. Snyder.: Bi-scale radiance transfer. In: ACM Transactions on Graphics, vol 22-3. (2003)
23. J. Stam.: Multiple Scattering as a Diffusion Process. In: Rendering Techniques, 41-50. (1995)
24. D. Wood, D. Azuma, W. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle.: Surface light fields for 3D photography. In: Proc. ACM SIGGRAPH. (2000)
25. X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, H.-Y. Shum.: Generalized displacement mapping. In:Proc. Eurographics Symposium on Rendering. (2004)
26. L. Wang, X. Wang, X. Tong, S. Hu, B. Guo, H.-Y. Shum.: View-dependent displacement mapping. In: Proc. ACM SIGGRAPH. (2003)

Ying Song Ying Song is a Ph.D candidate in Zhejiang University. She received her BS degree in the Computer Science department of Zhejiang University in 2001. Now she is a visiting student in MSRA.

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.

Xin Tong Dr. Tong is a researcher/project lead of the Internet Graphics group, Microsoft Research Asia. After receiving 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.

Stephen Lin Stephen Lin joined the Visual Computing Group of Microsoft Research, China in June 2000. He obtained his B.S.E. in electrical engineering from Princeton University, and then completed his Ph.D. in computer science and engineering at the University of Michigan shortly before joining MSRA.

Jiaoying Shi Jiaoying Shi is a professor of the Department of Computer Science and Engineering at Zhejiang University which is located in Hangzhou, Zhejiang Province of China. He is now the member of Academic Committee of State Key Lab of Computer Aided Design and Computer Graphics. He finished his School education in Ningbo, and graduated from Department of Physics at Leningrad University of USSR in 1960. Professor Shi is the Deputy Chairman of China Image and Graphics Association.

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 architectures. 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.