Real-Time Rendering of Translucent Objects with Variable Sizes

Chunhui Yao*†‡§, Bin Wang*†‡§
*School of Software, Tsinghua University, Beijing, China 100084
†Department of Computer Science and Technology, Tsinghua University, Beijing, China 100084
‡Key Laboratory for Information System Security, Ministry of Education, Beijing, China 100084
§Tsinghua National Laboratory for Information Science and Technology, Beijing, China 100084
ych08@mails.tsinghua.edu.cn, wangbins@tsinghua.edu.cn

Abstract

We present a novel technique for real-timely rendering translucent objects. Our technique is mainly based on translucent shadow maps with a new adaptive sampling strategy. In this sampling strategy, the hierarchy levels and positions for sampling are selected according to the size of target object. By our optimization, less storage in the texture is required than that in previous methods. Compared with previous methods, our technique provide a way to choose the proper sampling pattern and is applicable for a wider range of object sizes. With the implementation on GPU, the presented technique is able to render translucent objects in animated scenes where lights and material parameters vary real-timely without any lengthy preprocessing.

1. Introduction

Although new hardware environments and techniques appeared over the last few years, real-time photo-realistic image synthesis is still a challenging task. Rendering appearances of translucent materials is more difficult than rendering opaque objects for interactions between light and translucent materials are more complex. When arriving at a translucent surface, light will penetrate to the material, scatter inside and may exit at a different location and direction. This subsurface scattering causes the soft and smooth look of translucent materials, such as skin, marble and milk.

To render translucent materials, BSSRDF models are used. Jensen et al. [1] proposed an efficient BSSRDF model which is composed of exact single scattering [2] and multiple scattering by a dipole diffusion approximation. Jensen and Buhler [3] presented a two step approach to precompute and store irradiance samples and estimate the diffusion approximation using a hierarchical scheme. Then methods based on photon mapping [4] were presented, which can produce better results of translucent materials or participating media [5] [6]. These algorithms are more efficient, but not yet interactive rendering.

Several interactive methods based on Jensen's model have been presented. Lensch et al. [7] and Hao et al. [8] used precomputed information and estimated multiple scattering with vertex to vertex light transfer. However, both of them ignored single scattering which is necessary for physically-correct rendering. Wang et al. [9] presented a PRT-based method. It estimates both single and multiple scattering interactively, but requires a large amount of storage for precomputation. Furthermore, because of the expensive precomputation, the above methods only handle scenes with static objects.

Shadow mapping [10] is a commonly used technique to generate shadows in real-time applications. Many real-time rendering techniques are based on this technique [11] [12]. Dachsbacher and Stamminger [13] extended the shadow maps to store not only depth of objects but also irradiance and normal, which is called translucent shadow maps. By sampling translucent shadow maps with a fixed pattern, multiple scattering can be estimated in real-time without any preprocessing. Based on translucent shadow maps, Ki et al. [14] implemented the real-time estimation of single scattering.

Our technique focuses on rendering different appearances of translucent object in different sizes. Sizes are not carefully considered in previous real-time methods. However, in fact the variance in size is related to the appearances of translucent objects. Small objects tend to transmit more light and scatter less inside while large objects have light highly scattering. The different appearances are compared in Figure 1, where the two objets are lit by the same parallel light. The framework of our rendering algorithm is similar to the techniques of translucent shadow maps. Nevertheless, our algorithm has two advantages over the previous ones. First the storage of shadow maps is greatly reduced by eliminating the information of surface normal and irradiance. Second, instead of the fixed sampling pattern in previous methods [13], [14], we estimate multiple scattering with a different sampling strategy which is adaptive for different sizes. With the implementation on GPU, translucent objects in different sizes can be rendered in real-time with our technique.

The main idea of our algorithm is presented in Section 2. Section 3 and 4 respectively discuss the two important steps of the algorithm: shadow map generation and scattering estimation. In Section 3, we focus on our optimization to reduce the storage in textures. Section 4 mainly describes our adaptive sampling strategy for estimation of multiple

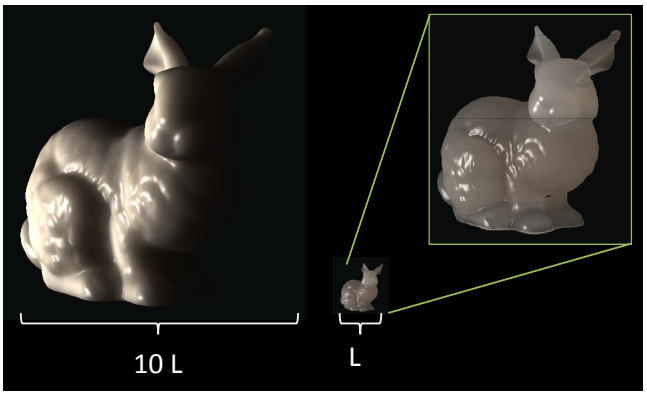

Figure 1. Appearances of translucent objects in different sizes. The left object is 10 times larger than the right one. Although they have the same shape and are both lit by the same parallel light, the appearances are different because of the variance of size.

scattering. Results by our technique are shown in Section 5. Finally we make a conclusion in Section 6.

2. Overview

Because of the subsurface scattering which is essential to translucent materials, the illumination result at a position on a translucent surface depends on the incident irradiance received at other positions nearby. Therefore irradiance samples over the surface are necessary for rendering translucent objects. For real-time rendering, we store the samples in a texture, which can be easily accomplished on GPU. We describe our rendering algorithm in Figure 2. The algorithm is composed of two passes: shadow map generation pass and rendering pass.

In the shadow map generation pass, shadow map textures with depth and irradiance information are generated by projecting the scene to the view space of the light. The projection resizes the object in image space to efficiently utilize the space of shadow map texture. By careful optimization, less information is stored in our shadow maps compared with previous methods.

In the rendering pass, the information in shadow maps is used for estimating subsurface scattering, including both single and multiple scattering. Diverse from previous methods, our shadow maps are sampled with an adaptive pattern for estimating multiple scattering according to the size of objects. This multiple scattering is combined with single scattering estimated with deterministic sampling.

3. Improved shadow map generation

The generation process of translucent shadow maps is similar to traditional methods: we render the scene in the

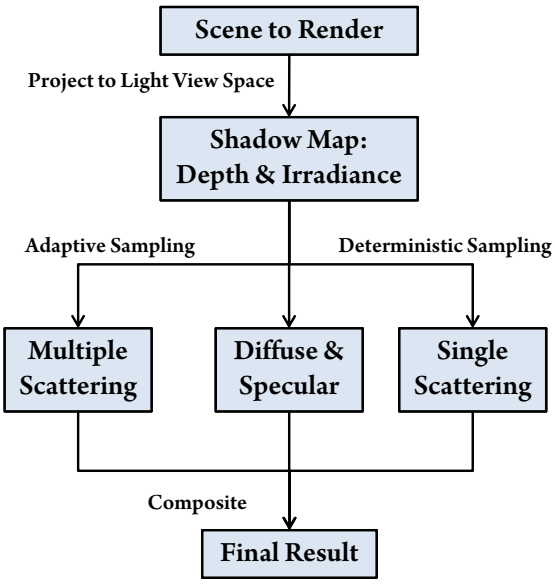

Figure 2. The flowchart of our rendering algorithm. A Shadow map texture is first generated by projecting the scene to the view space of light. Then the final result is produced by combination of the subsurface scattering estimated by sampling from the texture and diffuse/specular illumination computed by BRDF with a shadow test.

view of lights and store information in textures. However, our process is optimized in two ways: reducing the storage and efficiently utilizing the space.

For the estimation of subsurface scattering later, translucent shadow maps are supposed to contain irradiance information in addition to depth. Dachsbacher and Stamminger [13] put depth, irradiance and surface normal in their shadow maps which occupy five channels in the textures (one for depth, one for irradiance and three for normal). However, because the surface normal always appears in the form of dot product with incident direction in the expression of subsurface scattering, we store the result of dot product $(\vec{\omega}_i \cdot \vec{n})$ instead of the three components of normal to save two channels in textures. Irradiance information also can be eliminated by the following derivation. Irradiance received at a position on the surface is

$$E(x) = \int_{\Omega} L_i(x, \vec{\omega}_i)(\vec{\omega}_i \cdot \vec{n}) d\vec{\omega}_i. \quad (1)$$

For a single point or parallel light, all the incident radiance in a position comes along a certain direction $\vec{\omega}_i$. Additionally assuming the light has a constant intensity I_0 in this direction, the incident radiance received at the position x from a light at position x_0 is

$$L_i(x) = a(||x - x_0||)I_0, \quad (2)$$

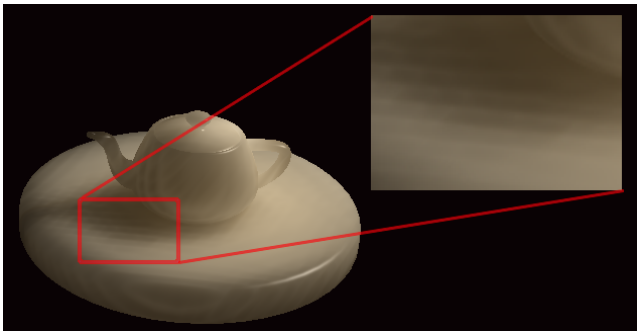

Figure 3. When the object is small, artifact appears because there are not enough pixels covered in the shadow map texture.

where a is attenuation term and it is usually expressed as $a(r) = r^{-2}$. If the light is a parallel light or a point light placed far away from the target, the attenuation almost does not change at different positions over the target object. As a result we can approximate it with a constant attenuation factor and merge it to the intensity. Then the radiance L_i is just equal to intensity I_0 and the expression of the irradiance becomes

$$E(x) = I_0 \cdot (\vec{\omega}_i \cdot \vec{n}). \tag{3}$$

Since the dot product $(\vec{\omega}_i \cdot \vec{n})$ has already been stored in the textures as the normal information, irradiance can be computed in the rendering phase. That means precomputed irradiance information is not necessary any more. In consequence, we store the depth value and the $(\vec{\omega}_i \cdot \vec{n})$ term in our shadow maps. There are only 2 components in our shadow maps compared with 5 components in traditional translucent shadow maps. Obviously this optimization suffers from the limitation of light types. But since we have already stored information in shadow maps which usually only handle simple light sources such as parallel lights or point lights, it does not bring much limitation for the application of this technique.

Because the target object may be in different sizes, the space of the shadow map texture is not effectively utilized when the object is small. It is a problem for the objects which cover small areas in the shadow maps, because incorrect results will appear when there are not enough pixels to sample, both in shadow and subsurface scattering, as shown in Figure 3. To eliminate this artifact, we project the target object to fill up texture. It is easy to implement by scaling height and width in the space of light view according to the size of the object, with the scaling ratio $k_s = L_t/L_0$ where L_t is the proper size to fill up the texture and L_0 is the original size of the target object.

Note that this resizing may cause the fixed sampling pattern not appropriate any more, because the scope to collect samples could be changed in the image space. To solve this problem, the following section describes a new adaptive sampling strategy which generates the sampling pattern according to the size of target object.

4. Scattering estimation

Texels in shadow maps are treated as sample points of irradiance. According to Jensen et al. [1], subsurface scattering is composed of single scattering and multiple scattering. For multiple scattering, a new adaptive sampling strategy is used. Single scattering is estimated by a deterministic sampling.

4.1. Adaptive sampling for multiple scattering

Because the influence of multiple scattering decreases exponentially with the distance [1], it is computed only with samples near the current position. But we still have to filter texels over a large area in the texture where influence of multiple scattering is significant enough. Since it is expensive to filter too many texels, mip-map levels are constructed to filter texels hierarchically.

The point on the object is first projected to the image space of shadow maps. We refer the projected point as "current point". The outgoing radiance of multiple scattering is computed by sampling texels in the shadow map texture. For the samples nearby, texels are selected in lower mip-map levels which provide detailed information. Samples far away from the current point are represented by texels in higher level mip-maps where each texel covers larger area in image space. Because we have projected the objects to the same size in the shadow maps, the same scope to collect samples in real world covers different areas in image space. The larger objects have smaller area to collect samples in shadow maps. Figure 4 presents the different areas in the image space for objects in different sizes.

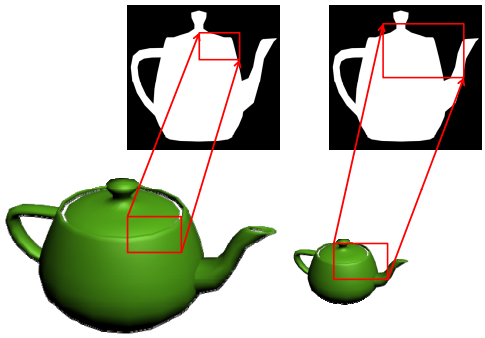

Figure 4. Because we have projected the objects to the same size in the image space, the scope in real world covers different areas in shadow maps for different object sizes.

As a result if the size of an object changes by $l' = kl_0$ where l_0 is the original size, the area covered in the image

space is $A' = A_0/k^2$ where A_0 is the original area in image space. Hence the sampling pattern need to be adapted for covering the new area. For simplicity, we first consider the case that only one texel is sampled. To cover the proper sample area, the hierarchy level of this texel need to be changed from N_0 to αN_0. Because in the n-th mip-map level one texel covers the area of 4^n texels in lowest level, the texel in the new level covers the area $A' = P_0 4^{\alpha N_0}$ where P_0 is the area represented by a texel in lowest level. By the equation of the two expressions of A', the factor α is derived as

$$\alpha = \frac{1}{N_0}\log_4\frac{A_0}{P_0} - \frac{1}{N_0}\log_2 k \tag{4}$$

which can be rewritten as

$$\alpha = C_1 - C_2 \log_2 k \tag{5}$$

where C_1 and C_2 are constants. The appropriate values of C_1 and C_2 can be chosen by experiments.

For a group of texels to be sampled, we adopt a similar approach that changes the hierarchy level of each texel from N_i to αN_i. Then we have our adaptive sampling strategy as following. We sample the shadow maps with a 7 x 7 grids where the current point is at the center. Each point x in the grids is mapped to a texel in the texture hierarchy. According to Equation 5, the level of mip-map is

$$n_s = \alpha K||x - x_0||, \tag{6}$$

where K is a factor for adjustment and x_0 is the position of current point. In practice, the mip-map level is rounded to the nearest integer from 0 to the largest level. Figure 5 shows two examples of the sampling pattern.

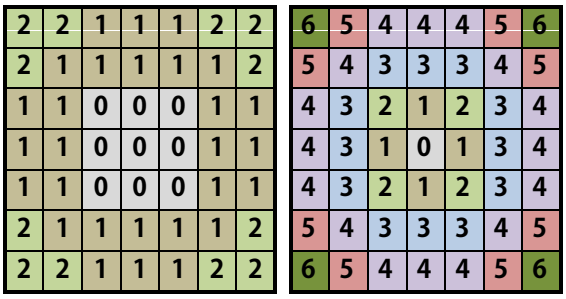

Figure 5. The 7 x 7 sampling patterns for two objects. The number in each grid indicates its mip-map level. The left pattern is for a large object. The level numbers are small so it covers a small area near the center point. In the right pattern for a small object, the level numbers grow rapidly from the center so it covers a large area in the texture.

We choose the center position of the sample as

$$x_s = x_0 + 2^{n_s}\Delta xy_{x-x_0} + \Delta z_x \tag{7}$$

with Δxy derived by texture coordinates offset and Δz derived by depth information in the shadow maps. The area represented by this sample is

$$A_s = 4^{n_s} P_0, \tag{8}$$

and the irradiance is

$$E_s = I_0 \cdot (\vec{\omega}_i \cdot \vec{n}), \tag{9}$$

with $(\vec{\omega}_i \cdot \vec{n})$ term in the shadow maps.

With the results of Equation 7, 8 and 9, the outgoing radiance of multiple scattering is computed as [3]:

$$L_i^{(d)}(x_o, \vec{\omega}_o) = F(x_o, \vec{\omega}_o)\sum R_d(||x_s - x_o||)E_s A_s \tag{10}$$

where R_d is the dipole approximation and F is a term related to outgoing position and normal. Because the value of F can be computed before sampling, the computation of R_d is the most expensive step. We can precompute the value of $R_d(r)$ with different r as a texture and perform texture lookups in the rendering phase.

4.2. Single scattering

Single scattering is sampled not over an area in the textures but along the refracted outgoing ray. Hence adaptive sampling is not necessary. Our single scattering is estimated with a Monte Carlo sampling method like the one used in [14]. We sample along the path of outgoing ray inside the material. For the purpose of sampling by importance, the number of samples reduces exponentially with distance. Therefore we sample the distance inside the object from the outgoing position by $S'_o = -\log(\xi)/\sigma_t$ where ξ is an evenly distributed random variable in (0, 1). Deterministic sampling [15] is used to reduce the number of samples. That means the distances are sampled fixed instead of choosing them randomly. For M samples required, each random variable is generated as $\xi_k = (k-0.5)/M$ for every integer k from 1 to M. The sample points in the outgoing path are projected to the space of shadow map textures. We sample texels around the projected point for the incident irradiance. The formula to compute outgoing radiance of single scattering is

$$L_i^{(1)}(x_o, \vec{\omega}_o) = \sigma_s F p(\vec{\omega}_i, \vec{\omega}_o)\frac{1}{M}\sum_{k=1}^{M}\frac{e^{(-s'_i + s'_{o,k})\sigma_t}}{pdf(s'_{o,k})}E_i. \tag{11}$$

See [14] for more details of Equation 11.

5. Implementation and results

We implemented our algorithm with Direct3D 9.0c and experiments on the graphics card of NVIDIA GeForce 9800 GT. We first render the scene to the shadow map texture. A texture format with two channels is enough, as we have only two components for each texel. Then we use the hardware

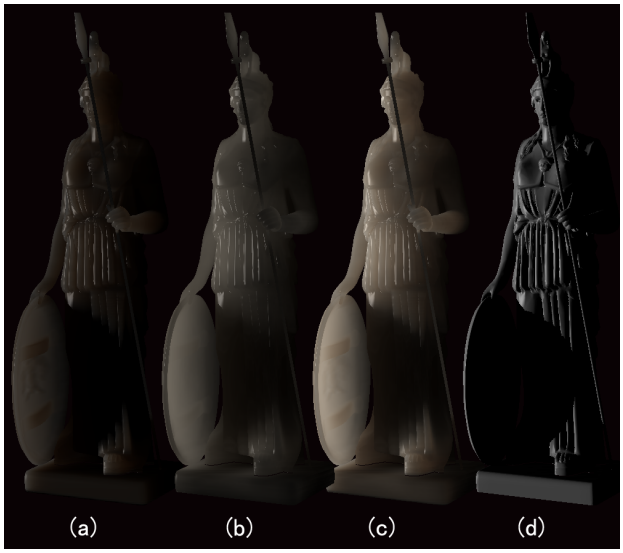

Figure 6. Illuminations with different kinds of scattering: (a) Single scattering; (b) Multiple scattering; (c) Full scattering; (d) Diffuse shading. Material parameters: $\sigma_s = [1.63\ 2.41\ 3.44]$, $\sigma_a = [0.0125\ 0.0206\ 0.0487]$, $\eta = 1.3$.

to automatically generate mip-map levels of the texture. The final result is produced by rendering the scene to the screen. Subsurface scattering is estimated on GPU along with diffuse and specular illumination. The target object is initially adapted to a proper size L_0 with a scaling transform. The size can be changed in real-time by a size factor k as $L = kL_0$.

Several examples produced by our implementation are presented here. We present the rendering results with different subsurface scattering components in Figure 6. Obviously objects in (a), (b) and (c) have smoother appearances compared with diffuse shading presented in (d). The combination of the two kinds of scatterings in (c) is the final result. These images are synthesized real-timely at a frame rate of about a hundred frames per second.

Our adaptive sampling is compared with other techniques in Figure 7 and 8. We put results of different techniques in different columns. The first two columns are produced real-timely (approximately 50 FPS in Figure 7 and 150 FPS in Figure 8). The left column is sampled with a fixed pattern for multiple scattering estimation, which is similar to that used in traditional translucent shadow maps [13]. The middle column is sampled by our adaptive strategy. Images in the right column are synthesized off-line in about 30 seconds by the technique presented by Jensen and Buhler [3]. We take the off-line results as standard of rendering quality. We compare them in each row where the parameters including size, material and light are the same. The size changes in different rows, while the material and light direction remain

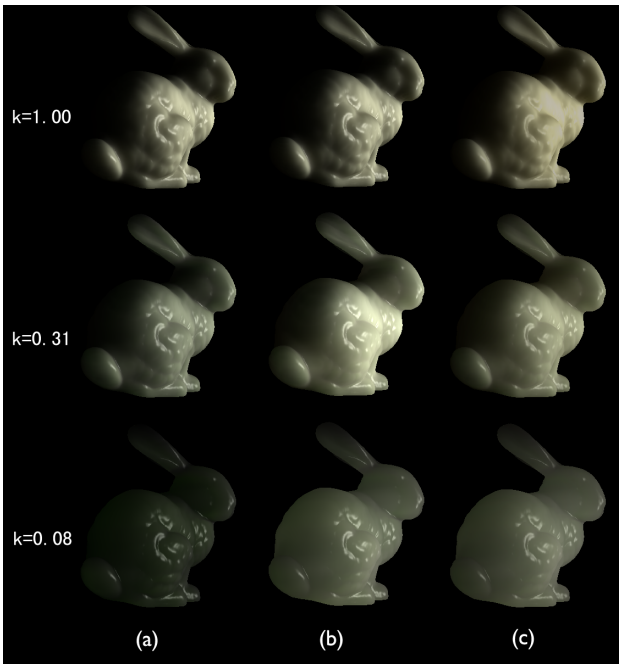

Figure 7. Rendering results of different techniques. The size decreases row-by-row and k is the size factor as $L = kL_0$. (a) Real-time rendering with a fixed sampling (50 FPS); (b) Real-time rendering with our adaptive sampling (50 FPS); (c) Off-line rendering (30 seconds). Our technique is much faster than off-line rendering with similar quality while the fixed sampling fails when the size is very small in the third row.

the same. There are no obvious differences between results of our technique and off-line rendering in different rows, although ours is much faster.

In Figure 7, the result with fixed sampling has few differences from ours when the object size is large in the first row, but in the last row where the size is greatly reduced, it looks much darker. The reason is that the sampling area is not large enough to cover the influence scope with the fixed pattern. In Figure 8, a different pattern is used for fixed sampling to achieve acceptable quality for small objects by manually changing the hierarchy levels. However, noise and incorrect illumination appears because the areas of the selected sample texels are too large to provide detailed information. These comparisons indicate that our adaptive sampling strategy is applicable for a wider range of sizes than fixed sampling.

We experiment for performance with models in different complexities. Our technique achieves high frame rates on models with tens of thousands of triangles. Table 1 presents the performance statistics. All the results are rendered in the resolution of 600 x 600 with a shadow map of 512 x 512. This table shows that our technique is applicable for real-time rendering.

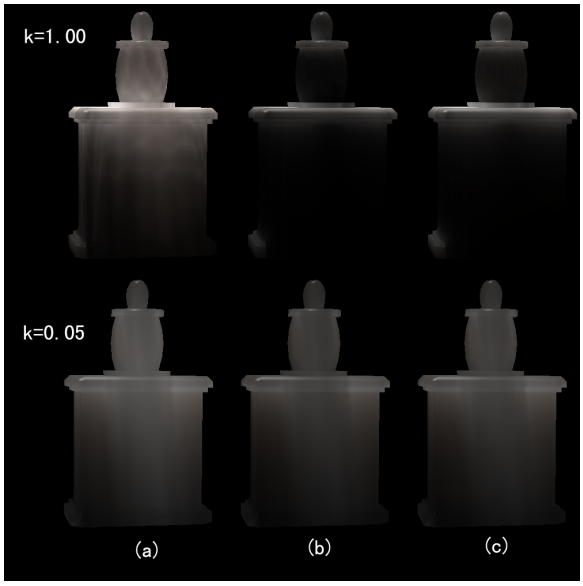

Figure 8. More comparisons. (a) Fixed sampling (150 FPS); (b) Ours (150 FPS); (c) Off-line (20 seconds). To produce acceptable results for small sizes, as in the second row, another fixed pattern is adopted in (a), but this time the fixed sampling fails for the large size in the first row.

Table 1. Frame rates of different models

Model	Triangles	Vertices	FPS
Stone	1114	802	255.3
Candle	12136	6936	142.5
Aphrodite	31398	15701	105.7
Bunny	69451	35947	53.1
Dragon	871414	437645	10.3

6. Conclusion

In this paper, we introduce a new technique to render translucent objects in real-time. Our technique is especially applicable for rendering the translucent appearances of different object sizes. With a new adaptive sampling strategy for estimating multiple scattering, we obtain a wider sphere of application. Because the whole rendering algorithm is implemented in GPU, we achieve high frame rates without any preprocessing.

Acknowledgement

This work is supported by National Science Foundation of China (Grant Nos. 90818011, 60773143 and 90715043), National High-Tech Research & Development Program of China (Grant No. 2007AA040401), National Basic Research Program of China (Grant No. 2004CB719400).

References

[1] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH '01*, pp. 511-518, 2001.

[2] P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of SIGGRAPH'93*, pp. 165-174,

[3] H. W. Jensen and J. Buhler. A rapid hierarchical rendering technique for translucent materials. In *Proceedings of SIGGRAPH'02*, pp. 576-581, 2002.

[4] H. W. Jensen. Global illumination using photon maps. In *Proceedings of the Seventh EurographicsWorkshop on Rendering*, pp. 21C30, 1996.

[5] H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of SIGGRAPH'98*, pp. 311-320, 1998.

[6] C. Donner and H. W. Jensen. Rendering translucent materials using photon diffusion. *SIGGRAPH'08 classes*, 2008.

[7] H. Lensch, M. Goesele, P. Bekaert, M. Kautz, J. Lang, and H.-P. Seidel. Interactive rendering of translucent objects. In *Proceedings of Pacific Graphics '02*, pp. 214-224, 2002.

[8] X. Hao, T. Baby, and A. Varshney. 2003. Interactive subsurface scattering for translucent meshes. In *Proceedings of 2003 ACM Symposium on Interactive 3D Graphics*, pp. 75-82, 2003.

[9] R. Wang, J. Tran, and D. Luebke. All-frequency interactive relighting of translucent objects with single and multiple scattering. In *Proceedings of SIGGRAPH '05*, pp. 1202-1207, 2005.

[10] L. Williams. Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH'78*, pp. 270-274, 1978.

[11] T. Annen, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz. Convolution shadow maps. In *Rendering Techniques 2007, volume 18 of Eurographics / ACM SIGGRAPH Symposium Proceedings*, pp. 51C60, 2007.

[12] C. Dachsbacher, M. Stamminger. Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pp. 203-231, 2005

[13] C. Dachsbacher and M. Stamminger. Translucent shadow maps. In *Proceedings of the Eurographics Workshop on Rendering*, pp. 197-201, 2003.

[14] H. Ki, J. Lyu, and K. Oh. Real-time approximate subsurface scattering on graphics hardware. *The 15th Pacific Conference on Computer Graphics and Applications*, pp. 403-406, 2007.

[15] A. Keller. Strictly deterministic sampling in computer graphics. *Technical report, mental images*, 2001.