A Real-time Subsurface Scattering Rendering Method for Dynamic Objects

LI Shuai, HAO Ai-min, WANG Zhen
State Key Laboratory of Virtual Reality Technology
Beihang University
Beijing, China
lishuaiouc@126.com

LIAN Ren-ming
Marine Hydrometeorological Centre
Chinese Navy
Beijing, China
lianrenming@163.com

***Abstract*—The paper proposes a real-time subsurface scattering rendering method for dynamic objects. Subsurface scattering owned by translucent objects is a complex optical phenomenon. Since the complexity of the property description of translucent materials and their interaction with light, the pre-computing methods are usually used to render subsurface scattering of translucent objects, but these approaches either limit the objects to be non-deformable or limit the translucent materials to be homogeneous. In this paper, A new translucent material describing and decomposed light computing model is set forth, on basis of which image-space approximate computing methods for single scattering and multiple scattering are respectively proposed. The Algorithm can fulfill the real-time rendering of dynamic objects by fully exploiting the capability of graphics processing units (GPU).**

***Index Terms*—subsurface scattering, dynamic scene, translucent material, real-time rendering, image space, GPU**

I. INTRODUCTION

Subsurface Scattering refers to the process of light transmission in translucent objects, of which light transmits into the object from a certain point on the surface, after internal scattering, finally transmits out from another surface point. Objects with such property usually have more mellow appearance, such as milk, jade, paraffin and biological skin and so on.

The history of studying on subsurface scattering is short. Ishmaru first gave a detailed optical analysis on translucent material in [1]. Later, many methods were proposed to carry out translucent objects rendering, such as finite element [2], the path tracing [3,4] and photon mapping [5], etc. Based on [1], Jensen proposed a dipole model [6] to approximately compute multiple scattering and achieved good rendering results. After that, he achieved a distinct capability promotion by employing hierarchical data structure [7]. Although the algorithm performance [7] improves a lot, it still needs a few seconds to render a simple scene. Then based on Jensen's method, Pellacini presented a hybrid rendering method, which mixed with Monte Carlo path tracing method and improved rendering efficiency by 4 times, but it is still an offline rendering method. Although PRT algorithms [8,9,10] can achieve real-time rendering for subsurface scattering, however, these algorithms usually limit the scenes to be static or partially dynamic. And these methods generally need several hours or even more than a dozen hours to carry out pre-computing, and the pre-computed results consume too much memory and video card memory. So it is difficult to use them in dynamic or large-scale scenes.

Besides, among the algorithms putting main objective on real-time rendering, Green's method [11] is simplest. It amended diffuse reflection function used in real-time rendering and made the function getting positive results when the dot metrix of the light vector and the surface normal vector is negative. So, the surface opposite to light source can also get a certain degree of light, which represented the light intensity permeated from the translucent objects. And Borshukov simulated the skin subsurface scattering of animals or human beings by simply blurring the rendering result of direct lighting [12]. Although these methods [11, 12] are all lack of physical reality, they still earn a certain amount of living space because of their low computing cost. Contrasting to [11, 12], some image space based methods [13,14] can get better real time results, they first create float textures to store geometric information of the scene by multi-pass rendering, and then make approximate calculation for multiple scattering item in image space using Jensen's dipole model. But such image space based methods can only simulate the low-frequency scattering effect and deal with homogeneous materials.

To sum up, current real-time rendering approaches for subsurface scattering generally have some kinds of limitation: (1) limiting the properties of materials to be homogeneous (2) limiting the scenes to be static due to pre-computation (3) only simply simulating the subsurface scattering effects at a certain degree but fully ignoring the physical reality. However, the approach in this paper can realize real-time rendering of approximate global light effects for dynamic translucent materials. The main innovation of this paper mainly includes 3 points. First, it introduces a translucent materials describing model which can deal with simple non-homogeneous materials. Second, it proposes a decomposed light computing model for translucent materials based on analyzing the process of physical light transmission. Third, based on the above two points, it brings forward an image-space approximate computing approach, which can achieve real-time subsurface scattering rendering for dynamic translucent objects. Therefore, the approach proposed in this paper is important for enhancing the reality of real-time virtual reality applications.

II. Translucent Material Describing Model

The space variability of scattering properties for materials is generally associated with the physical structure of the materials, for example, skin is composed of lipid layer, epidermis layer, dermis layer and tissue layer from outside to inside. The use of layered physical structure can simplify the describing complexity of the material scattering properties.

As shown in Fig1, we present a multi-layer model for describing translucent material property. The model mainly describes the thickness, absorption coefficient and scattering coefficient. For a certain point V on the object with surface normal $\vec{N}_i$, the corresponding thickness of different material property layer can be expressed by the transmitting distance along the opposite normal direction. And it's assumed that each layer has same absorption coefficient and scattering coefficient. So, the material scattering coefficients can be described by $n \times 3$ float values, which represents RGB components having different parameters in n layers.

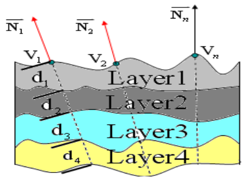

Figure 1. Translucent material describing model

We use floating-point texture to store the layered material properties by expanding the meaning of common color texture, of which each texel has four 32-bit float values for RGBA channels, and respectively represents the depth of different layer. The color texture coordinates of the three-dimensional models can be used to index material property texture, and $n\%4$ floating textures are needed to store n layer depth. Similarly, for heavy variable heterogeneous materials, more floating textures should be used to store the absorption coefficients and scattering coefficients of those materials. Here, each texture can only store a certain type of coefficient in one layer, in which each texel channel respectively represents a certain coefficient corresponding to RGB components.

$TexDep_i$ $TexAbs_i$ and $TexScat_i$ represent the i-th thickness texture, absorption coefficient texture and scattering coefficient texture respectively, Ω represents the surface of the three-dimensional model, and d_k, α_k, σ_k respectively refer to the k-th layer thickness, RGB absorption coefficient vector and RGB scattering coefficient vector, so the formula is achieved as follows:

$$\begin{aligned} &\forall v \in \Omega,\ d_k = Sample(TexDep_i, t_{uv})_{.m} \\ &\alpha_k = Sample(TexAbs_i, t_{uv}) \\ &\sigma_k = Sample(TexScat_i, t_{uv}) \end{aligned} \quad (1)$$

Of which, t_{uv} refers t to the color texture coordinate corresponding to point v, $Sample$ means texture sample function, and $i = k\%4, m = k - i \times 4$.

III. Decomposed Light Computing Model

$$L_o(x_o, w_o) = \int_A \int_{2\pi} S(x_i, w_i, x_o, w_o) L_i(x_i, w_i)(n \cdot w_i) d\omega_i dA(x_i) \quad (2)$$

Formula(2) describes the translucent objects lighting model based on BSSRDF, of which $L_o(x_o, w_o)$ represents the output light intensity from w_o direction, at x_o location, from w_i direction, and $L_i(x_i, w_i)$ represents the input light intensity from w_i direction, at x_i location. If given a BSSRDF function, output light intensity can be gotten by the considering all the incident light on the object, but it is hard to compute the BSSRDF. Even Monte Carlo method is employed to carry out approximate calculation, the time cost is still very expensive.

Considering the interaction between light and translucent objects, this paper proposes a decomposed light computing model. As shown in Fig2, To the incident lights that never enter into the translucent object, similar to local light computing model, we divide it into diffuse reflection light (L_{id})and reflection light(L_{is}); to the incident lights from inner surface, we divides them into three types, refraction light(L_{ir}),single scattering irradiance of partly high-frequency light source(L_{iss}), and multiple scattering irradiance of low-frequency environment light source(L_{ims}), of which L_{iss} and L_{ims} are closely related to the material properties of the objects, whose computing methods will be introduced below in detail.

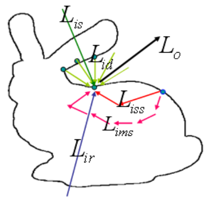

Figure 2. decomposed light computing model

We can get the approximate lighting result by summing above five light components using formula (3).

$$\begin{aligned} L_i(x_i, w_i) \approx\ & \lambda_1 L_{id}(x_i, w_i) + \lambda_2 L_{is}(x_i, w_i) + \\ & \lambda_3 L_{it}(x_i, w_i) + \lambda_4 L_{iss}(x_i, w_i) + \lambda_5 L_{ims}(x_i, w_i) \end{aligned} \quad (3)$$

After decomposing incident lights, we employ different methods to deal with different types of light components. L_{is} and L_{it} can be computed simply using local lighting model, L_{id} can be computed using the screen space ambient occlusion method, L_{iss} can be computed in image space using deferred shading method, and L_{ims} can also be approximately computed in image space by introducing pixel lights.

IV. IMAGE-SPACE APPROXIMATE COMPUTING

A. *Single Scattering Approximate Computing*

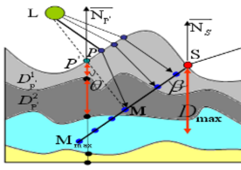

Figure 3. Single subsurface scattering computing

As shown in Fig3, for a certain surface vertex S with normal vector $\overrightarrow{N_s}$, $\overrightarrow{SE}$ is the view direction. The light from the inner position M should contribute to the outputting light intensity of S due to single subsurface scattering. And the position of M can be calculated by formula (4):

$$M_i = S + d_{si} \times \overrightarrow{T_S}, \overrightarrow{T_S} = \text{Refract}(\overrightarrow{N_S}, \overrightarrow{SE}) \quad (4)$$

$\overrightarrow{T_S}$ is the refraction direction, d_{si} is the distance transmitting into the object. When light enters the object, single subsurface scattering can be executed within a certain depth; but if beyond the depth, light will execute multiple subsurface scattering. So for a particular material, we set a maximum single subsurface scattering depth $D_{\max}$, which means the maximum depth should be considered for single subsurface scattering along the opposite surface normal direction. The $M_{\max}$ corresponding to the vertex can be computed by formula (5):

$$d_{si} = D_{\max} / \cos\beta = D_{\max} / (\overrightarrow{N_S} \cdot \overrightarrow{T_S}) \quad (5)$$

Therefore through accumulating the intensity from all the position M, we can get the single subsurface scattering intensity for S. The calculation formula is as follows:

$$L(S, \omega_{out}) = \int_S^{M_{\max}} L_{ss}(M, \omega_{out}^{\overrightarrow{T_S}}) dM \approx \sum_S^{M_{\max}} L_{ss}(M, \omega_{out}^{\overrightarrow{T_S}}) \quad (6)$$

$L_{ss}(M, \omega_{out}^{\overrightarrow{T_s}})$ shows the contribution of light intensity for single subsurface scattering from S to M, and $\omega_{out} = \overrightarrow{SE}, \omega_{out}^{\overrightarrow{T_s}} = \overrightarrow{T_s}$, whose calculation formula is as follows:

$$L_{ss}(M, \omega_{out}^{\overrightarrow{T_S}}) = Q(M, \omega_{out}^{\overrightarrow{T_S}}) e^{\int_S^M -\sigma_t(s) ds} \quad (7)$$

$e^{\int_S^M -\sigma_t(s) ds}$ shows the attenuation from M to S, $\sigma_t(s)$ is the absorption coefficient, $Q(M, \omega_{out}^{\overrightarrow{T_s}})$ is the exiting light along the direction $\overrightarrow{T_s}$ from M. The calculation formula is as follows:

$$Q(M, \omega_{out}^{\overrightarrow{T_S}}) = \sigma_s(M) p(M, \omega_{out}^{\overrightarrow{T_S}}, \omega_{in}^{\overrightarrow{T_P}}) L_{ia}(M, \omega_{in}^{\overrightarrow{T_P}}) \quad (8)$$

$\sigma_s(M)$ is the scattering coefficient at position M, $p(M, \omega_{out}^{\overrightarrow{T_s}}, \omega_{in}^{\overrightarrow{T_P}})$ is the phase function to describe the scattering ratio of the light intensity from ω_{in} to M along the direction $\omega_{out}^{\overrightarrow{T_s}}$, whose calculation formula is as follows.

$$p(M, \omega_{out}^{\overrightarrow{T_S}}, \omega_{in}^{\overrightarrow{T_P}}) \approx \frac{1-g^2}{4\pi(1+g\cos\theta)^2} = \frac{1-g^2}{4\pi(1+g(\overrightarrow{PM} \cdot \overrightarrow{MS}))^2} \quad (9)$$

$L_{ia}(M, \omega_{in}^{\overrightarrow{T_P}})$ is the attenuated light intensity which arrives at M along the direction of ω_{in}, whose calculation formula is as follows .

$$L_{ia}(M, \omega_{in}^{\overrightarrow{T_P}}) = L_{is}(P, \omega_{in}) \prod_{j=1}^{MaxLayer} e^{-\sigma_t^j d_j} \approx L_{is}(P', \overrightarrow{LP'}) \prod_{j=1}^{MaxLayer} e^{-\sigma_t^j d_j} \quad (10)$$

As shown in Fig3, $\omega_{in}^{\overrightarrow{T_P}} = \overrightarrow{PM}$ $\omega_{in} = \overrightarrow{LP}$, $\overrightarrow{PM}$ is the refraction direction of incident light $\overrightarrow{LP}$. $\prod_{j=1}^{MaxLayer} e^{-\sigma_t^j d_j}$ expresses the residual ratio after material absorbing, σ_t^j is the material scattering coefficient of j-th layer, d_j is the transmission distance through the j-th layer; $MaxLayer$ is the number of layers transmitting from P to M. $L_{is}(P, \omega_{in})$ is the incident light intensity of high-frequency point light source at P. In the implementation, we use P' rather than P to make approximate calculation. The calculation formula is as follows:

$$d_j \approx \min((\| \overrightarrow{P'M} \| - \sum_{i=0}^{j-1} \frac{D_i}{\overrightarrow{N_{P'}} \bullet \overrightarrow{LM}}), \frac{D_j}{\overrightarrow{N_{P'}} \bullet \overrightarrow{LM}}) \quad (11)$$

D_j denotes the depth of the j-th layer corresponding to P', we can get D_j and $L_{is}(P', \overrightarrow{LP'})$ by sampling textures.

B. *Multiple Scattering Approximate Computing*

In Jenson's dipole Model for multiple scattering [6], incident light at any position on the object surface can be replaced by a pair of point lights. Among them, the point light above the object surface is known as virtual light source, while the other is known as real light source.

Expressed in this approximation, the relationship between the radiant existence $M_o(x_o)$ from x_o and irradiance flux $\Phi_i(x_i)$ from x_i can be set up as follows:

$$dM_o(x_o) = d\Phi_i(x_i) \frac{\alpha'}{4\pi} \left(C_1 \frac{e^{-\sigma_{tr} d_r}}{d_r^2} + C_2 \frac{e^{-\sigma_{tr} d_v}}{d_v^2} \right) \quad (12)$$
$$C_1 = z_r \left(\sigma_{tr} + \frac{1}{d_r} \right), \quad C_2 = z_v \left(\sigma_{tr} + \frac{1}{d_v} \right)$$

In formula (12), $\alpha' = \sigma_s' / \sigma_t'$ is the weakened reflection rate; $\sigma_{tr} = \sqrt{3\sigma_a \sigma_t'}$ is the transmission disappearing coefficient; $d_r = \sqrt{r^2 + z_r^2}$ is the distance from the exiting point to the real point light; $d_v = \sqrt{r^2 + z_v^2}$ is the distance from the exiting point to the virtual point light; $r = \|x_o - x_i\|$ is the distance between incident point and exiting point; $z_r = l_u$ and $z_v = l_u(1+4/3A)$ are respectively the distance from two point lights to the object's surface. σ_s' is the weakened scattering coefficient, and σ_t' is the weakened absorption coefficient. σ_s' and σ_t' can be calculated by formula $\sigma_t' = \sigma_s' + \sigma_a, \sigma_s' = \sigma_s(1-g)$, of which σ_s is the original scattering coefficient and σ_a is the absorption coefficient. In general, light will tend to be isotropy ($g=0$) after several multiple scattering. $l_u = 1/\sigma_t'$ is called Mean-Free Path.

$A = (1+F_{dr})/(1-F_{dr})$ can correct the error brought by different refractive coefficient on the media boundary, in which F_{dr} is Fresenel item that can be seen as a function about relative refractive coefficient: $F_{dr} = -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta$.

In Jenson's dipole model, mean-free path equals the distance from the real point light to the object's surface, so for multiple scattering of heterogeneous material, we only consider the first layer scattering and make use of image space based method to approximately compute on GPU. To calculate multiple scattering needs diffuse irradiance texture and render a full-screen quadrilateral. Here are the steps:

(1) Make each pixel (P) in diffuse irradiance texture as a point light source. The surrounding pixel lights of P may contribute to the ultimate exiting light intensity of P by the multiple scattering. The contribution will decrease quickly as the distance between P and the pixel light become larger. Therefore, we only sample 12 pixel lights (each with light intensity $L(S_i)$) closest to P to compute multiple scattering.

(2) Indexing deferred shading texture to get the position coordinates for the object surface vertexes corresponding to P and S_i , and then calculate the distance (r_i) between sampling vertex and current vertex;

(3) Respectively calculate C_1^i , C_2^i , d_r^i , d_v^i and relevant coefficient on the basis of r_i, and use formula (13) to get the multiple scattering exiting light intensity $L(P)$ by summing the contribution of all the sampling pixel lights.

$$L(P) = \sum_{i=1}^{12} L(S_i) \frac{\alpha'}{4\pi} \left(C_1^i \frac{e^{-\sigma_{tr} d_r^i}}{(d_r^i)^2} + C_2^i \frac{e^{-\sigma_{tr} d_v^i}}{(d_v^i)^2} \right) \quad (13)$$

V. Experimental Results

This paper employs a PC with Pentium 4 3.0GHz CPU, 1G memory, NVIDIA GeForce 8800 GTS 512M video card and Windows XP operating system as experimental platform. Experimental program bases on DirectX 9.0c API, and shader program is compiled using Shader Model 3.0 profile.

Fig4 gives the synthetic rendering result of goddess model. In this figure, （A） and （B） respectively represent single subsurface scattering and multiple subsurface scattering; (C) is the final rendering result by synthesizing kinds of light components using formula (3). The parameter combination is $(\lambda_1,\lambda_2,\lambda_3,\lambda_4,\lambda_5) = (0.3, 0.3, 0.05, 0.05, 0.3)$. The scattering coefficients and absorption coefficients used for the goddess model are approximate to the coefficients of apple material. From Fig4 (C), we can see that the final rendering result is more mellow and the scattering features in thinner regions of the goddess model seems more obvious , such as wrist, nose and ears. Fig5 gives our experimental results for deformed frog model, in which pictures (A),(B),(C),(D) are respectively the rendering results in the 1st ,15th ,30th and 45th frames.

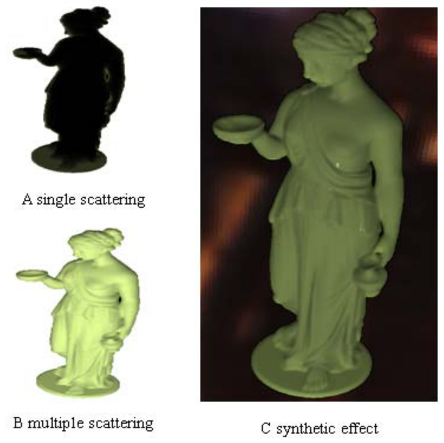

Figure 4. Synthetic rendering

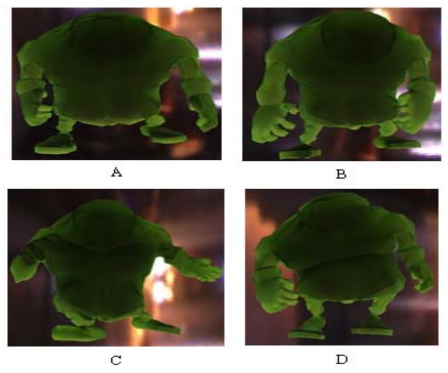

Figure 5. Deformable objects

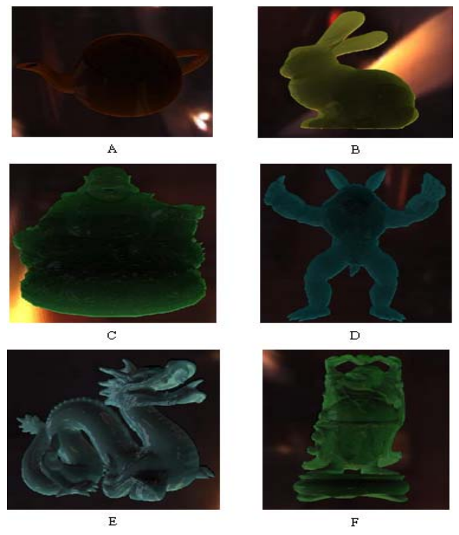

Figure 6. Moveable rigid objects

Fig6 provides the rendering results of the moveable rigid objects. (A), (B), (C), (D), (E), and (F) are separately corresponding to teapot, bunny, Maitreya, monster, and Buddha models, whose complexity increase step by step. From the rendering result, we can see single scattering components due to high-frequency point light take a large proportion in picture(A), (B), (C), (F),which look like the material of jade, for example, teapot's spout and cover, bunny's ear and tail, maitreya' s head and buddha's skirt. These areas express the permeability of jade well. The permeability has tendency to decrease from the margins to the centers of the models, which inosculates with the visual characteristics of translucent materials. However, pictures (D) and (F) weaken the overall permeability, whose visual characteristics seems closer to glass and porcelain. It is mainly obtained by adjusting the proportion value for various kinds of light components in formula (3). By experiment, we can adjust the parameter value $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5)$ to simulate translucent materials with a variety of different surface properties.

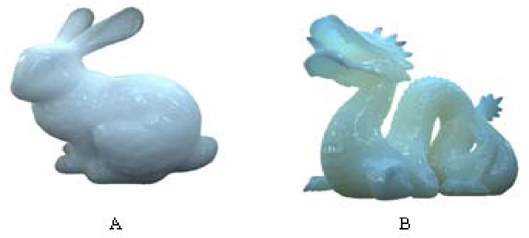

Figure 7. Reference image based on PRT algorithm

In general, PRT algorithms first need to pre-process the scenes based on physics law. For complex scene with hundreds of thousands of triangles, the pre-processing time usually needs several hours. And at running time, the pre-computation data is loaded into RAM and Video RAM to do real-time rendering. So it is well known that the results rendered by PRT algorithms can be seen as absolute right in physics. Therefore, we also realize the PRT-based subsurface scattering rendering algorithm and list its rendering results in Fig7. Comparing with the reference images, we can find that our method can get good rendering results which is approximate to the image rendered based on physics. In addition, our method can make up the deficiency of PRT algorithm and achieve real-time subsurface scattering rendering for dynamic scenes without any pre-computation.

Table 1 Experiment Data

Objects	Triangles	FPS		Scattering Coefficients			Absorption Coefficients		
		512	1024	First Layer	Second Layer	Third Layer	First Layer	Second Layer	Third Layer
deformed frog	2,900	231	71	2.29	2.39	1.97	0.0030	0.0034	0.046
teapot	6,400	230	70	0.74	0.88	1.01	0.032	0.17	0.18
goddess	63,932	208	68	2.29	2.39	1.97	0.0030	0.0034	0.046
bunny	69,451	205	66	0.15	0.21	0.38	0.015	0.077	0.19
maitreya	86,994	185	63	1.09	1.59	1.79	0.013	0.070	0.145
dragon	100,000	178	59	0.70	1.22	1.90	0.0014	0.0025	0.0142
monster	345,944	124	52	2.29	2.39	1.97	0.0030	0.0034	0.046
Happy buddha	1,084,450	63	42	1.09	1.59	1.79	0.013	0.070	0.145

Table 1 states the number of triangles of each three-dimension model used in this experiment and the corresponding scattering coefficient, absorption coefficient and diffuse color, etc. The scattering coefficients and absorption coefficients originate from [6]. But [6] only presents scattering parameter of homogeneous materials after analyzing the real material sample. To prove the algorithm's validity, the multiple scattering parameters listed in this paper are gotten by making a little adjustment. In experiments, we use two resolutions to create kinds of dynamic textures, and the rendering FPS can be found respectively in the third column in table1.

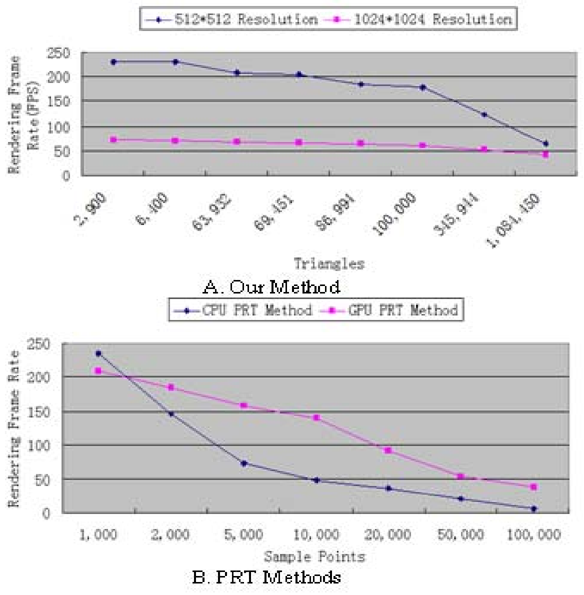

Figure 8. Efficiency analysis

Fig8(A) presents two efficiency curves which are respectively gotten under the resolutions 512*512 and 1024*1024 for dynamic textures. In Fig8(A), we can see that when the resolution is 512*512, the algorithm has higher rendering efficiency, especially for the complex scene with one million triangles, whose rendering FPS can reach up to more than 60 frame per second. But when the resolution changes to 1024*1024, the algorithm efficiency declines a lot, especially for the simple scene, its efficiency may decline to a quarter of the original. This ratio essentially inosculates with the ratio between the dynamic texture resolutions, that is 512*512/1024*1024=1/4. And to Buddha model with 1,080,000 triangles, whose rendering FPS can reach over 40 frames, this proves that our method can guarantee real-time rendering well. In addition, in Fig8(A) booth curves look stable and cannot constitute inverse ratio between the FPS and the number of the scene triangles. Meanwhile, the red curve looks more stable than the blue curve. All these shown that the algorithm efficiency is greatly influenced by the dynamic texture resolution rather than the complexity of model. In general, when the resolution is improved, the ultimate rendering quality will also improve a lot. So according to the practical needs, the balance between rendering efficiency and quality can be made by choosing reasonable resolution for the dynamic textures.

Fig8(B) presents two efficiency curves, which are respectively corresponding to CPU based PRT algorithm and GPU based PRT algorithm. It is shown that whether using CPU or GPU, efficiency curves almost constitute inverse ratio with the number of PRT sampling vertexes. PRT algorithm usually choose the vertexes of three-dimension models as sampling points in the period of pre-computation, so, the efficiency of PRT algorithm has great relation with the complexity of the scene. Shown in Fig8(B), even for GPU based PRT algorithm, when the number of sampling points reaches 10,000, the rendering efficiency is only about 30 FPS. Furthermore, it need cost several hours to pre-process the scene and the pre-computed data will cost too much RAM memory. So, PRT algorithm is hard to satisfy the real-time rendering of static complex scenes or dynamic scenes due to pre-computation. After all, our algorithm has obvious advantages from the view of algorithm efficiency and algorithm applicability.

VI. Conclusion and Future Work

This paper proposes an Image Space Based real-time rendering Algorithm for dynamic Translucent objects. And the algorithm can deal with simple heterogeneous translucent materials. The experiments indicate that our method is practical and has obvious advantages in efficiency. It can achieve relatively good rendering results which is approximate to the image rendered by physics based algorithm.

Besides, our algorithm also has some work to do in the future. First, the algorithm may result in aliasing in high-frequency area such as object's margin and high curved areas when the resolution of dynamic texture is lower. Although this problem can be restrained by raising resolution of dynamic texture, it need relatively higher cost. So in the following jobs, we will first find an effective anti-aliasing method.

Second, we will focus on a self-adaptive method to control the weighted parameters in formula (3), so that we can make the simulation more realistic and vivid for some special materials.

Acknowledgment

The work is supported by the National Natural Science Foundation of China under Grant No. 60533070 and the National High-Tech Research and Development Plan of China 863 under Grant No. 2007AA01Z337.

References

[1] Ishimaru A. Wave Propagation and Scattering in Random Media, Volume 1[M]. New York: Academic Press, 1978

[2] Franc¸ois X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. IEEE Transactions on Visualization and Computer Graphics,1(3):240–254, 1995.

[3] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In Proceedings of ACM SIGGRAPH1993, pages 165–174, 1993.

[4] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In Proceedings of Eurographics Rendering Workshop 1996, pages 91–100,1996.

[5] Henrik Wann Jensen and Per Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In Proceedings of ACM SIGGRAPH 1998, pages 311–320, 1998.

[6] Jensen H. W., Marschner S. R., Levoy M., et al. A practical model for subsurface light transport[A]. SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques[C]. New York: ACM Press, 2001: 511-518

[7] Jensen H. W., Buhler J. A rapid hierarchical rendering technique for translucent materials[A]. SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques[C]. New York: ACM Press, 2002: 576-581

[8] Xuejun Hao and Amitabh Varshney. Real-time rendering of translucent meshes. ACM Transactions on Graphics, 23(2):120–142, April 2004.

[9] Rui Wang, John Tran, and David Luebke. Allfrequency interactive relighting of translucent objects with single and multiple scattering. ACM Transactions on Graphics, 24(3):1202–1207, August 2005.

[10] Peter-Pike Sloan, Ben Luna, and John Snyder.Local, deformable precomputed radiance transfer. ACM Transactions on Graphics,24(3):1216–1224, 2005.

[11] Green S. Real-time approximations to subsurface scattering[A]. GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics[C]. Massachusetts: Addison-Wesley, 2004: 263-278

[12] Borshukov G., Lewis J. P. Realistic human face rendering for "the matrix reloaded"[A]. SIGGRAPH '03:ACM SIGGRAPH 2003 Sketches & Applications[C]. New York: ACM Press, 2003: 1-1

[13] Hyunwoo Ki, Jihye Lyu, Kyoungsu Oh. Real-Time Approximate Subsurface Scattering on Graphics Hardware

[14] Eugene d'Eon, David Luebke. Advanced Techniques for Realistic Real-Time Skin Rendering. GPU Gems3, Addison Wesley Professional.,August 02, 2007.