Real-Time and Interactive Rendering for Translucent Materials Such as Human Skin

Hiroyuki Kubo[1], Yoshinori Dobashi[2], and Shigeo Morishima[1]

[1] Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo, Japan
[2] Hokkaido University, Kita 14, Nishi 9, Kita-ku, Hokkaido, Japan
hkubo@suou.waseda.jp,
doba@ime.ist.hokudai.ac.jp,
shigeo@waseda.jp

Abstract. To synthesize a realistic human animation using computer graphics, it is necessary to simulate subsurface scattering inside a human skin. We have developed a curvature-dependent reflectance functions (CDRF) which mimics the presence of a subsurface scattering effect. In this approach, we provide only a single parameter that represents the intensity of incident light scattering in a translucent material. We implemented our algorithm as a hardware-accelerated real-time renderer with a HLSL pixel shader. This approach is easily implementable on the GPU and does not require any complicated pre-processing and multi-pass rendering as is often the case in this area of research.

Keywords: computer graphics, real-time rendering, subsurface scattering.

1 Introduction

Simulating sub-surface scattering is one of the most important issues to realistically synthesize translucent materials and has received a great deal of attention from the rendering research community over a decade.

We propose a curvature-dependent reflectance functions (CDRF) that mimic the presence of a translucent material which is dominated by higher-order scattering events such as human skin, marble, jade, and so on. Since the proposed function is a local illumination model, the method synthesizes realistic translucent materials in real-time.

The features of our method are as follows.

- computationally lightweight (almost the same cost as Lambert shading.)
- ingle-pass rendering (saving computation cost and graphics memory.)
- unnecessary mesh parameterization (can be applied various objects compared with [1].)
- can be easily used to stylize subsurface scattering effects.

We focus on the issue which is relatively local scale scattering, such that the incident light hit the surface, and bleed into a narrow area. Although, our method does not consider an exact scattering in a whole shape of an object, we propose a plausible approach using local illumination approximation.

G. Salvendy, M.J. Smith (Eds.): Human Interface, Part II, HCII 2011, LNCS 6772, pp. 388–395, 2011.

A fast approximation of subsurface scattering has been receiving a great deal of attention from game developers and others. Our method is straightforward for artists to control and much simpler than Kolchin's [2] described below.

2 Previous Work

Rendering translucent materials has been an interesting research for a decade. There are currently two types of approaches to synthesize these objects, one is for off-line rendering and the other is for real-time rendering.

2.1 Offline Rendering

The photon mapping [3, 4, 5] is capable to simulate light transport inside a translucent material accurately, though, it is also known that the calculation cost is notably expensive. Since it basically requires over several hours to synthesize a single image, the photon mapping is not practical for real-time rendering.

The method developed by Jensen et al. [6, 7] improved significantly on the speed of the simulation. Using a bidirectional surface scattering distribution function (BSSRDF) model, they combine a dipole diffusion approximation with single scattering computation, yet still cannot produce real-time rendering.

2.2 Real-Time Rendering

In 2007, d'Eon et al. [1, 8] developed a subsurface scattering rendering method using texture-space diffusion technique. They approximate the profile of multi-pole model using sum-of-Gaussians formulation, then apply gauss filtering of varying radius of blur to an irradiance map in the texture space. They realize a high quality, gpu accelerated method, however, it also contains two issues that cannot be avoided easily. One is the calculation cost and the size of required memory. Compared with the latest CPU and graphics processing unit, current consumer gaming devices are not able to compute very fast, and do not equip enough memory. Their method requires more than ten rendering passes (1-pass for rendering irradiance texture, 12-passes for 6 gaussian convolution, 1-pass for stretching and 1-pass for composit), thus, the method is computationally expensive and requires huge capacity of graphics memory. The other problem is the necessity of mesh parameterization. Since their technique approximates subsurface scattering by blurring irradiance in the texture space, every vertices of the model are necessary to be parameterized. Mesh parameterization of arbitrary shape is still challenging issue. Therefore, their technique is not very much practical for current consumer gaming devices.

Kolchin[2] developed a curvature-based shading method. The basic idea behind this paper is similar to that of Kolchin's, in that the effects of subsurface scattering depend on curvature. Curvature of a surface can be used to derive a local-illumination approximation of subsurface scattering.

In this paper, we describe that our method is significantly simpler and easier to control than Kolchin's method. Compared with these works, we propose a much faster technique than Jensen's Dipole Model and d'Eon's texture space diffusion, and a significantly simpler and easier to control than that of Kolchin's.

3 Our Approach

According to the previous works [6, 7], we realize that the effects of subsurface scattering tend to be more noticeable on small, intricate objects than on simpler, flatter ones, which indicates that surface complexity largely determines these effects. For the purposes of our research, we decided to use curvature to represent surface complexity, combined with a simple local illumination model.

In this paper, we describe a curvature-dependent reflectance function (CDRF) for synthesizing translucent materials. CDRF is not a global illumination model such as BSSRDF, but a local illumination model. Compared with BRDF which is typical local illumination model, CDRF depends on not only incident direction $\boldsymbol{\omega}_i$ but also surface curvature κ.

In most cases of this kind of research, there are often too many parameters for artists to control subsurface scattering and stylize the appearance of translucent materials. Instead, to compute radiance on a surface using CDRF, there is only one parameter σ_0 that represents the degree of light scattering inside a material. σ_0 is not only provided by curve-fitting to simulated data-set, but also manipulated by an artist.

Furthermore, this is easy implementable on the GPU and doesn't require any complicated pre-processing and multi-pass rendering as is often the case in this area of research [9].

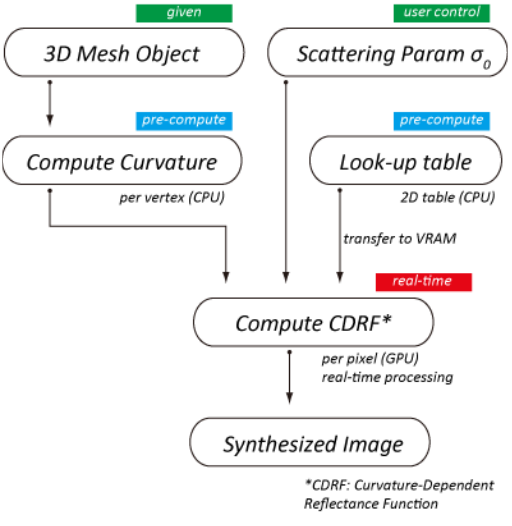


Fig. 1. Workflow

To synthesize an image of translucent materials, we used the following procedures shown on figure 1. The input is 3D mesh object, and the output is a synthesized image. Prior to rendering, we acquire surface curvature of the mesh on CPU computing proposed by [10, 11]. The value of CDRF depends on curvature κ, scattering parameter

σ_0 and $\omega_i \cdot n$. We pre-compute it for all combinations of these parameters and store it in a 2-dimensional look-up table (lut). Then, using this lut for faster computation, we calculate CDRF in pixel shader to render translucent materials in real-time.

4 Curvature-Dependent Reflectance Function

The effects of subsurface scattering tend to be more noticeable on smaller, more intricate objects than on simpler, flatter ones. This seems to indicate that surface complexity largely determines these effects. For the purposes of our research, we decided to use curvature to represent surface complexity, combined with a simple local illumination model.

To represent subsurface scattering effect, we propose curvature-dependent reflectance function (CDRF) $f_r^c(\theta_i, \kappa)$. CDRF is defined by convolution of incident light energy $E(\theta_i)$ and a gauss function $g(\theta_i, \sigma)$ for providing blurring effect.

$$f_r^c(\theta_i, \kappa) = (E * g)(\theta_i) \tag{1}$$

where,

$$E(\theta_i) = \max(\cos\theta_i, 0) \tag{2}$$

$$g(\theta_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\theta_i^2}{2\sigma^2}\right) \tag{3}$$

Accordingly, observing on θ_i axis, σ is supposed to be, relatively, in inverse proportion to radius r. Therefore, we assume that

$$\sigma(\kappa) = \frac{\sigma_0}{r} = \sigma_0 \kappa \tag{4}$$

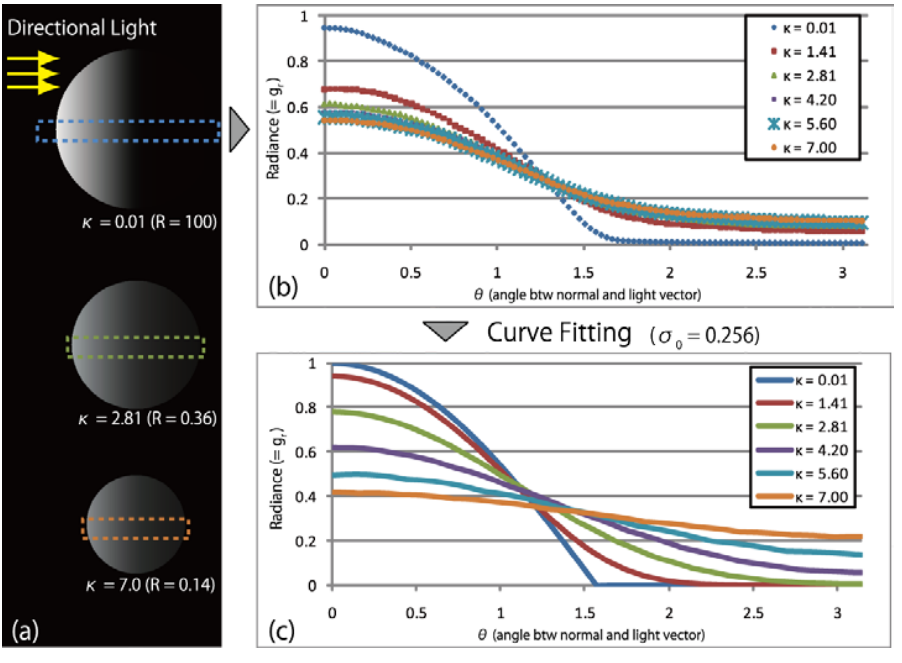


Fig. 2. Curve fitting

σ_0 corresponds to the scattering intensity in the unit sphere. Using this reflectance function, diffuse radiance L_d on a surface is calculated as following equation,

$$L_d = \int_\Omega k_d f_r^c(\theta_i, \kappa) L_i(\boldsymbol{\omega}_i) d\boldsymbol{\omega}_i \quad (5)$$

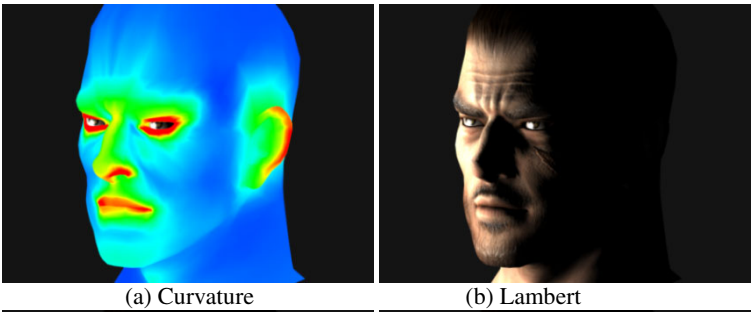
Note that, $L_i(\boldsymbol{\omega}_i)$ is an intensity of incident light from direction $\boldsymbol{\omega}_i$, k_d is diffuse albedo, and Ω is over all the incident light directions.

To confirm the validity of the formula, Eq(1), we rendered several spheres of varying radii to reveal the relationship between curvature and radiance using the photon tracing. The spheres are illuminated by a directional light from the left side of the sphere, as shown in Figure 2-(a).

The calculated color of each pixel on the equatorial line represents a relation of light angle and radiance of the sphere's particular curvature (Figure 2-(b)). Then, we fit a curve to the obtained data using CDRF formula Eq. (2). The curves are not exactly the same, but it is similar and well fit. This means that CDRF approximate formula we propose is roughly accurate. In terms of the way to determine the scattering parameterσ_0, it is not only provided by curve-fitting to simulated data-set, but also manipulated by an artist.

5 Results

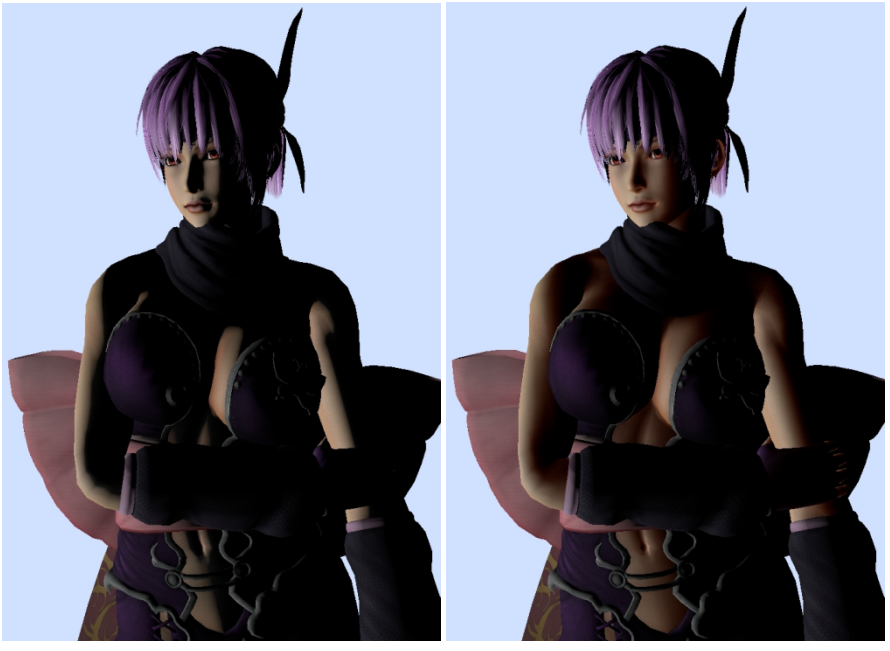
The results are summarized in figure3-4. These images are obtained from our implementation running on a 3.06GHz Intel® Core™ 2 Duo CPU with NVIDIA® GeForce® 9300M GS GPU. We implemented our algorithm as a hardware-accelerated real-time renderer. The renderer is implemented as a pixel shader in HLSL.



(a) Curvature (b) Lambert

(c) Texture Blur Method (d) Our Method

Fig. 3. An example of human head



(a) Lambert (b) Our Method

Fig. 4. an example of human skin

To validate our algorithm, we have tested our algorithm using a human head model. Figure3-(a) displays surface curvature of the model, hot colors represent higher curvature, and vice versa. For comparison, figure3-(b) and (e) are rendered using Lambertian, figure3-(c) and (f) are rendered using texture blurring method[1]. Figure3-(d) and (g) are synthesized images using our CDRF. Compared with the Lambertian, we are able to synthesize more natural, softer shading effects especially around its nose and cheek. The appearance of ours (figure3-(d),(g)) may be not very effective than that of texture blurring method (figure3-(c),(f)), though, the advantage of our method is calculation cost and unnecessary of mesh parameterization.

Compared with Lambert shading, additional costs to compute CDRF are only 3-times texture sampling and a few arithmetic operations, thus the frame rate of our method is not significantly increased. Since our method is implemented as a single-pass shader, according to our implementation and computational environment, over 10 times faster than texture blurring method which requires over 10-passes.

Furthermore, we rendered another example of a human skin as shown in figure4. Compared with the hard appearance of the image on the left, our method is able to synthesize soft appearance of the skin.

6 Conclusion and Discussion

We have developed a method for approximating the effects of subsurface scattering using a curvature-dependent reflectance function. Since the function is a local illumination model, we are able to synthesize realistic translucent materials in real-time. Furthermore, our system can be easily used to stylize subsurface scattering effects because only one parameter σ_0 is required.

We approximate an object surface locally as a sphere's surface of corresponding radii according to the curvature. Thus, we cannot consider global object shape such as object thickness. Therefore, compared with a real object, difference of appearance tends to be more noticeable, when the object is especially thin such as leaves and papers. However, our method can be applied almost all practical scene, and it is possible to synthesize translucent materials plausibly.

In our future work, we will apply our techniques to deformable objects by computing curvature on the GPU.

Acknowledgement. We appreciate the feedback offered by TECMO KOEI GAMES CO., LTD.

References

1. d'Eon, E., Luebke, D., Enderton, E.: Efficient rendering of human skin. In: Rendering Techniques 2007: 18th Eurographics Workshop on Rendering, June 2007, pp. 147–158 (2007)
2. Kolchin, K.: Curvature-based shading of translucent materials, such as human skin. In: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, GRAPHITE 2007, pp. 239–242. ACM, New York (2007)
3. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, SIGGRAPH 1986, pp. 143–150. ACM, New York (1986)
4. Dorsey, J., Edelman, A., Jensen, H.W., Legakis, J., Pedersen, H.K.: Modeling and rendering of weathered stone. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, SIGGRAPH 1999, pp. 225–234. ACM Press/Addison-Wesley Publishing Co, New York, NY, USA (1999)
5. Jensen, H.W., Legakis, J., Dorsey, J.: Rendering of wet materials. In: Rendering Techniques 1999, pp. 273–282 (1999)
6. Jensen, H.W., Marschner, S.R., Levoy, M., Hanrahan, P.: A practical model for subsurface light transport. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH 2001, pp. 511–518. ACM, New York (2001)
7. Jensen, H.W., Buhler, J.: A rapid hierarchical rendering technique for translucent materials. ACM Trans. Graph. 21, 576–581 (2002)

8. Nguyen, H.: GPU Gems 3. Addison-Wesley Professional, Reading (2007)
9. Mertens, T., Kautz, J., Bekaert, P., Seidelz, H.-P., Reeth, F.V.: Interactive rendering of translucent deformable objects. In: Proceedings of the 14th Eurographics workshop on Rendering, EGRW 2003, pp. 130–140. Aire-la- Ville, Switzerland (2003)
10. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential–geometry operators for triangulated 2-manifolds. In: Visualization and Mathematics III, pp. 35–57 (2003)
11. Goldfeather, J., Interrante, V.: A novel cubic-order algorithm for approximating principal direction vectors. ACM Trans. Graph. 23, 45–63 (2004)