Real-Time Rendering of Heterogeneous Translucent Materials with Dynamic Programming

Tadahiro Ozawa[1] and Midori Okamoto[1] and Hiroyuki Kubo[2] and Shigeo Morishima[3]

[1]Waseda University, Japan [2]Nara Institute of Science and Technology, Japan [3]Waseda Research Institute for Science and Engineering

Abstract

Subsurface scattering is important to express translucent materials such as skin, marble and so on realistically. However, rendering translucent materials in real-time is challenging, since calculating subsurface light transport requires large computational cost. In this paper, we present a novel algorithm to render heterogeneous translucent materials using Dijkstra's Algorithm. Our two main ideas are follows: The first is fast construction of the graph by solid voxelization. The second is voxel shading like initialization of the graph. From these ideas, we obtain maximum contribution of emitted light on whole surface in single calculation. We realize real-time rendering of animated heterogeneous translucent objects with simple compute and our approach does not require any precomputation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Real-time rendering approach provides photo-realistic images and important to create contents with presence. A lot of techniques of real-time rendering have been proposed. However, technique for translucent materials remains challenging task, because it takes high cost to simulate subsurface scattering (SSS). Representing the SSS is significant to synthesize photo-realistic images. To realize fast rendering of translucent materials, many previous works handle SSS approximately. Jensen et al. [JMLH01] introduce *Dipole model*, one of *BSSRDF*, considering only multiple scattering effects. Applying that *BSSRDF*, Jensen synthesizes photo-realistic images, but rendering time is lowered to minutes and this model can not be applied to heterogeneous materials. Then volumetric-based approaches, e.g, tetrahedron structure [WWH*10], deliver high quality rendering automatically, but even latest algorithm are not fast enough for rendering deformable objects. In contrast, there is some works that represents the spread of the light transport using blurred texture [BL05]. Those approaches realize real-time rendering of translucent materials. However, blurring in texture space does not always exhibit geometrically correct light transport, since the distance in texture space does not always mean spatial distance. Those methods find it difficult to handle optically thin objects.

In this paper, we present a novel technique to handle SSS considering most significant optical path, while using a common volumetric data - voxelization approaches. We realize real-time rendering of highly scattering materials such as skin, marble. Our result in Figure 1 shows rendering of heterogeneous materials including deformed objects. Applying solid voxelization, our method can handle arbitrary deformation such as destruction. In addition, our approach handles light transport not in texture space or screen space but in spatial coordinate space. Therefore, our method can be applied to, for instance, optically thin objects. Our method delivers visually plausible result to off-line rendering such as path tracing.

2. Rendering algorithm

When an object which is made of highly scattering materials is illuminated by only single spotlight which irradiation area is microscopic enough, the radiance on the surface is determined by the shortest optical path length to the incident point. In this case, radiance is easily calculated by the use of Dijkstra's Algorithm [Dij59]. It is one of dynamic programing (DP) and the algorithm to calculate shortest path length between all nodes in the graph. We assume that general light source can be regarded as a collection of light which includes various intensity and we extend the above approach to general light source. In particular, we initialize goal nodes to intensity of each light rays. In other words, we use *weighted goal nodes* instead of dealing with whole goal nodes equally. This signifies that, for each point, we determine the radiance taking account of only single ray which has maximum contribution to the point.

Our approach is based on a two-step algorithm. First, we obtain the path length between incident and exit surface points. Applying Dijkstra's Algorithm, we create and initialize a graph appropriate for the objects. Second, we estimate the radiance from node's value.

DOI: 10.2312/egp.20161038

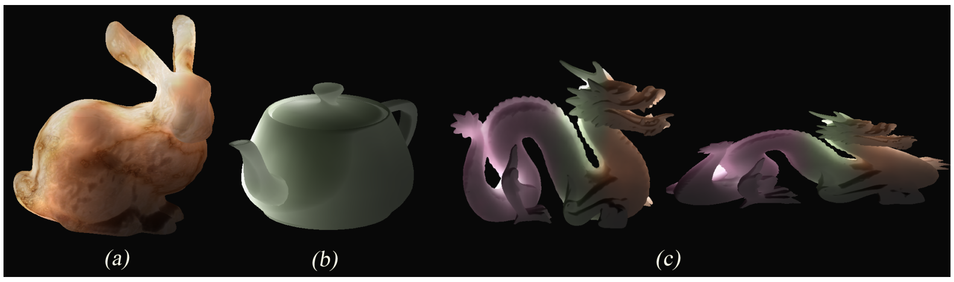

Figure 1: *Stanford Bunny (143fps), Teapot (165fps) and Dragon(167fps)*

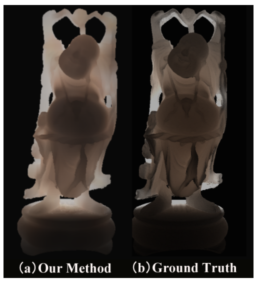

Figure 2: *Comparison with Ground Truth*

The radiance of emitted light L is then:

$$L = F_t(\eta)\exp(-n(v_i)) \tag{1}$$

where $n(v_i)$ is value node v_i has and this value represents optical path length, F_t is Fresnel transmittance term.

2.1. Creation and initialization of the graph

We apply solid voxelization to create a graph for Dijkstra's Algorithm. The graph that nodes are arranged in a grid pattern is constructed creating each node in the center of voxel. Next, edge is created and each edge connects neighboring 26 voxels We apply extinction coefficient to the edge's cost. Thereby makes node's value shortest optical path length to the incident point. Then we initialize each node. In this context, points to render correspond to a start node and points illuminated correspond to a goal node. This initialization is similar to shading of voxels. Therefore, we identify which area is illuminated to initialize the each goal node properly. To initialize graph, we apply four-channel $reflective\ shadow\ map$ (RSM) [DS05] that the world position and intensity are off-screen rendered. A pixel in the RSM represents a small light source. Scanning all pixel in RSM, we shade goal nodes. The initial value $n_0(v_i)$that node v_i has is then:

$$n_0(v_i) = -\ln(F_t(\eta)P) \tag{2}$$

$$P = \frac{1}{N^2}\sum_i A_i \tag{3}$$

where $F_t(\eta)P$ represents the amount of light on the voxel, F_t is Fresnel transmittance term and η is reflection index. A_i is power of small light source which is included in voxel v_i and which value is obtained from RSM. The number of small light source depends on the resolution of RSM. Therefore, the total power is divided by the square of N, resolution ratio of voxel and RSM,

2.2. Radiance calculation

From the volume rendering equation, if we ignore the in-scattering and emission, radiance is calculated from Equation 1. Radiance is calculated only in voxel units. The resolution of voxel is too low to synthesize high quality images. Therefore, we create a 3D texture of the same resolution as the voxel and interpolate the voxel's color.

3. Result

Figure 1 shows the result of our method for various shape and material. (a) is *Bunny* made of homogeneous translucent material with marble texture and *Bunny* is illuminated from the left. (b) is *Teapot* made of homogeneous material and illuminated from the left back. (c) is undeformed and deformed heterogeneous *Dragon* illuminated from the back. Our method can handle animated objects like (c) conducting voxelization at each frame. We have implemented our algorithm on an Intel Core i7-5960X 3.00GHz with 64.0GB memory and NVIDIA Geforce GTX 980 with 4GB graphics memory. We also compared our result against a ground truth image that was computed using path tracing. Figure 2 shows comparison. (a) is our result and (b) is ground truth. In Figure 2, we render a homogeneous marble *Buddha* illuminated from the back. It takes five hours to synthesize the image of path tracing. On the other hand, rendering speed of our method is 5.5ms and realize faster rendering. Our method handles subsurface scattering correctly and synthesizes image visually similar to ground truth.

4. Conclusion and future works

We present a method to render the heterogeneous translucent materials applying Dijkstra's Algorithm to the uniform voxel grid graph. We realize real-time rendering of animated heterogeneous materials. We synthesize plausible images. In future work, we also evaluate our approximation using rendering equation.

References

[BL05] Borshukov G., Lewis J. P.: Realistic human face rendering for the matrix reloaded. In *ACM Siggraph 2005 Courses* (2005), ACM, p. 13. 1

[Dij59] Dijkstra E. W.: A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271. 1

[DS05] Dachsbacher C., Stamminger M.: Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (2005), ACM, pp. 203–231. 2

[JMLH01] 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* (2001), ACM, pp. 511–518. 1

[WWH*10] Wang Y., Wang J., Holzschuch N., Subr K., Yong J.-H., Guo B.: Real-time rendering of heterogeneous translucent objects with arbitrary shapes. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 497–506. 1