Pacific Graphics (2011)
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin (Editors)

Real-time Rendering of Endless Cloud Animation

K. Iwasaki[1], T. Nishino[1] and Y. Dobashi[2]

[1]Wakayama University, Japan, [2]Hokkaido University, Japan

Abstract

In this paper, we propose a real-time animation method for dynamic clouds illuminated by sunlight and skylight with multiple scattering. In order to create animations of outdoor scenes, it is necessary to render time-varying dynamic clouds. However, the simulation and the radiance calculation of dynamic clouds are computationally expensive. In order to address this problem, we propose an efficient method to create endless animations of dynamic clouds. The proposed method prepares a database of dynamic clouds consisting of a finite number of volume data. Using this database, volume data for the endless animation is generated at run time using the concept of Video Textures, and this data is rendered in real-time using GPU.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Realistic animation of dynamic clouds is an essential component for depicting outdoor scenes, and several applications, such as flight simulators, landscape simulators, and games, require real-time rendering of dynamic clouds as animated backgrounds. For such applications, the creation of endless cloud animations is beneficial. In order to render realistic clouds, the scattering of sunlight and skylight as well as multiple scattering of light must be taken into consideration. However, these calculations are computationally expensive. Previous rendering methods [DKY*00, HL01, SKSU05, BNM*08] are limited to static clouds or do not take into account skylight or multiple scattering of light in order to achieve real-time performance. Real-time animation of clouds illuminated by sunlight and skylight with multiple scattering still remains a challenging problem.

To address this problem, we propose an efficient method to create an endless animation of dynamic clouds. In the proposed method, a finite number of successive volume data, called basic volume data, are created and stored in a database as a preprocessing step. The basic volume data are created by using one of the traditional dynamic cloud simulation methods.

At run time, new volume data are created by selecting and blending two volume data from the database based on the concept of video textures [SSSE00]. The proposed method then renders the volume data illuminated by sunlight and skylight, taking into account multiple scattering of light. Although multiple scattering of light and skylight are important for realistic image synthesis, the calculations of these components are computationally expensive. The main difficulty lies in the calculation of transmittances, which requires integration of the densities. In order to overcome this difficulty, the proposed method precomputes the transmittances from the sky and from the inside of clouds for each basic volume data in the database, and the transmittances of the volume data created at run time are efficiently calculated at run time using the precomputed transmittances. The proposed GPU implementation demonstrates real-time rendering of dynamic clouds illuminated by sunlight and skylight with single anisotropic scattering and multiple isotropic scattering.

2. Generation of Dynamic Volume Data

This section describes a method to create volume data based on the concept of video textures [SSSE00], which can create an endless video from the input video. Video textures define the similarities between every pairs of frames of an input video by calculating differences in the colors of the two frames. The proposed method extends the concept of video textures to volume data. The proposed method precomputes successive volume data $V_b(i)(i = 1, 2, \cdots, N_{vol})$ representing dynamic clouds and stores these data as basic volume data.

DOI: 10.2312/PE/PG/PG2011short/073-076

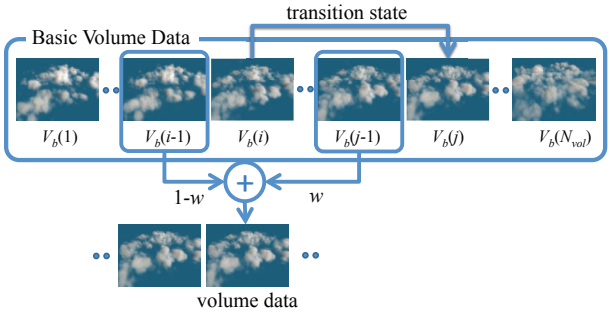

Figure 1: *Generating new volume data using basic volume data.*

An animation of clouds is created by stochastically selecting similar basic volumes, and rendering these volumes. The similarity D_{kl} between two volumes $V_b(k)$ and $V_b(l)$ is calculated by: $D_{kl} = \sum_{n=1}^{N_{voxel}} || \ _k(x_n) - \ _l(x_n)||^2$, where x_n represents a voxel, N_{voxel} is the number of voxels for the basic volume data, and $\ _k$ and $\ _l$ are the densities for $V_b(k)$ and $V_b(l)$, respectively. The proposed method stochastically selects basic volume data $V_b(l)$, subsequent to $V_b(k)$, with high similarity D_{kl}. The probability of selection P_{kl} is calculated by: $P_{kl} = \frac{\exp(-D_{k+1,l}/\ )}{\sum_k \exp(-D_{k+1,l}/\ )}$, where $\ $ is a parameter specified by the user.

At run time, the volume data subsequent to $V_b(k)$ is selected according to a random number obeying P_{kl}. To realize a smooth transition between the two volumes $V_b(k)$ and $V_b(l)$, the proposed method uses a cross dissolve technique for a certain interval H. That is, volume data V generated at run time for a certain frame is represented by a linear interpolation of two basic volumes, $V_b(i)$ and $V_b(j)$, where $i = k-H+1,\cdots,k$ and $j = l-H+1,\cdots,l$. More specifically, density $\ $ at voxel x of the volume data V is represented by: $\ (x) = w \ _i(x) + (1.0-w) \ _j(x)$, where w is a weight to realize the cross dissolve effect and it changes from zero to one.

3. Rendering

The proposed method handles sunlight and skylight for the incident illumination. Although our method basically assumes the scattering property to be isotropic for the computational efficiency, our method can handle anisotropic scattering for single scattering. We first describe the radiance calculation method for the single scattering, followed by that for the multiple scattering.

3.1. Single Scattering

The radiance of single scattered light $L^1(x,\Omega_o)$ at voxel x along direction Ω_o is calculated by:

$$L^1(x,\Omega_o) \ = \ L^1_{sun}(x,\Omega_o) + L^1_{sky}(x,\Omega_o), \qquad (1)$$

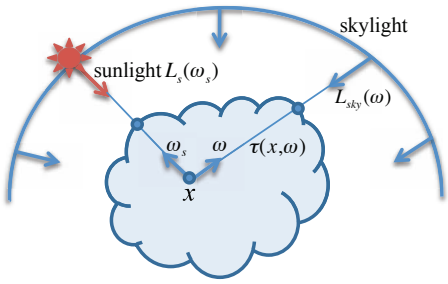

Figure 2: *Radiance calculation of clouds illuminated by sunlight and skylight.*

$$L^1_{sun}(x,\Omega_o) \ = \ (\ x)\frac{\Omega}{4}L_s(\Omega_s)\ (x,\Omega_s)p(\Omega_s,\Omega_o), \qquad (2)$$

$$L^1_{sky}(x,\Omega_o) \ = \ (\ x)\frac{\Omega}{4}\int_S L_{sky}(\Omega)\ (x,\Omega)p(\Omega,\Omega_o)d\Omega, \qquad (3)$$

where $\ $ is the density, Ω is the albedo, $L_s(\Omega_s)$ is the radiance of sunlight, p is the phase function, S represents a set of directions on the unit sphere, L_{sky} is the radiance of skylight, and $\ (x,\Omega)$ is the transmittance from x to infinity in the Ω direction, which is calculated by

$$\ (x,\Omega) = \exp(-\ _t\int_0^\infty \ (\ x+u\Omega)du), \qquad (4)$$

where $\ _t$ is the extinction coefficient. $L^1_{sun}(x,\Omega_o)$ in Eq. (2) is the scattered radiance of sunlight and can be calculated easily by casting a ray from each voxel to the sun. On the other hand, the calculation of the scattered radiance of skylight $L^1_{sky}(x,\Omega_o)$ is computationally expensive, because this calculation requires the calculation of transmittances for many directions and the integration of $L_{sky}(\Omega)\ (x,\Omega)p(\Omega,\Omega_o)$ over S.

In order to efficiently calculate $L^1_{sky}(x,\Omega_o)$, the proposed method calculates $L^1_{sky}(x,\Omega_o)$ in SH domain similar to [ZRL*08] as:

$$L^1_{sky}(x,\Omega_o) = \ (\ x)\frac{\Omega}{4}\left[(\mathbf{L}_{sky} * \ (\ x)) \star \mathbf{p}\right]\cdot\mathbf{y}(\Omega_o), \qquad (5)$$

where $\mathbf{L}_{sky} = (l_0, l_1, \cdots, l_{m^2-1})$ is a vector of SH coefficients for $L_{sky}(\Omega)$ and calculated by projecting $L_{sky}(\Omega)$ onto SH basis $y_m(\Omega)$, that is, $l_m = \int_\Omega L_{sky}(\Omega)y_m(\Omega)d\Omega$. $\ (x)$ and $\mathbf{p}$ are also vectors of SH coefficients for $\ (x,\Omega)$, and p respectively. $*$ and $\star$ represent SH triple product and convolution operators described in [SKS02], and $\mathbf{y}(\Omega_o)$ is a vector of m-th order SH functions. In the following, we focus on the calculation of $\ $.

To calculate $\ $, the proposed method uses the SHEXP technique [RWS*06]. At each voxel of the i-th basic volume data $V_b(i)$, the proposed method precomputes the optical depth $f^i_b(x,\Omega) = \int_0^\infty \ _i(x+u\Omega)du$. Then $f^i_b(x,\Omega)$ is projected onto the SH basis and is represented as $f^i_b(x,\Omega) = \mathbf{f}^i_b(x)\cdot\mathbf{y}(\Omega)$, where $\mathbf{f}^i_b(x)$ is a vector of SH coefficients for $f^i_b(x,\Omega)$.

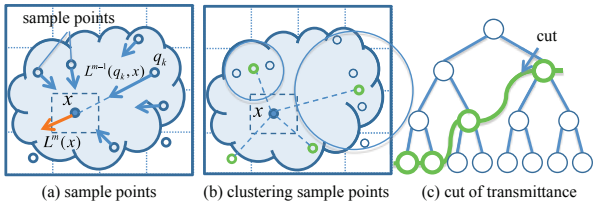

Figure 3: *Calculation of multiple scattering using sample points and cuts.*

As described in the previous section, volume data V are created at each frame by blending two basic volume data. The density (x) for V is represented by a weighted sum of the densities $_i(x)$ and $_j(x)$. In this case, the optical depth $f(x,\Omega)$ for V is also represented by the weighted sum of the optical depths of the two basic volumes:

$$f(x,\Omega) = w f_b^i(x,\Omega) + (1-w) f_b^j(x,\Omega), \tag{6}$$

where f_b^i and f_b^j are the optical depths for $_i$ and $_j$, respectively. Therefore, the coefficient vector $\mathbf{f}$ for $f(x,\Omega)$ can be easily calculated using the precomputed vectors $\mathbf{f}_b^i$ and $\mathbf{f}_b^j$ as $\mathbf{f} = w\mathbf{f}_b^i + (1-w)\mathbf{f}_b^j$. The SH vector (x) for transmittance $(x,\Omega) = \exp(-\ _t f(x,\Omega))$ is calculated by the SHEXP operation [RWS*06] of $\mathbf{f}$: $(x) = \exp_*(-\ _t\mathbf{f})$, where $\exp_*$ is the SHEXP operation and calculated by:

$$\exp_*(\mathbf{f}) \approx \exp\left(\frac{f_0}{\sqrt{4}}\right)\left(a(||\hat{\mathbf{f}}||)\cdot\mathbf{1} + b(||\hat{\mathbf{f}}||)\cdot\hat{\mathbf{f}}\right), \tag{7}$$

where $\hat{\mathbf{f}} = (0, f_1, f_2, \cdots, f_{m^2-1})$ and $\mathbf{1} = (\sqrt{4}, 0, \cdots, 0)$. a and b are tabulated functions of the magnitude of input vector $\hat{\mathbf{f}}$. To calculate the tables for a and b, our method samples a number of voxels from the basic volume data and calculates SH vectors of optical depths and corresponding SH vectors of transmittances for those sampled voxels. Then a and b are calculated by applying the least square method to these vectors. Using this method, transmittance (x,Ω) for the new volume data V can be efficiently calculated.

3.2. Multiple Scattering

In order to calculate the multiple scattering of light, the proposed method generates many sample points in the volume (see Fig. 3(a)). Sample points are uniformly distributed in the volume. The radiance of multiple scattered light at each voxel is calculated by accumulating the scattered light at sample points. The radiance of the n-th scattered light at voxel x, $L^n(x)$, is calculated by:

$$L^n(x) = (x)\frac{\Omega}{4}\sum_{k=1}^{K} L^{n-1}(q_k)\,(q_k,x)G(q_k,x), \tag{8}$$

where K is the number of sample points, L^{n-1} is the radiance of the $(n-1)$-th scattered light at sample point q_k, is the transmittance between x and q_k and is calculated by $(q_k,x) = \exp(-\ _t\int_x^{q_k} (y)dy)$, and G is the geometric term [WFA*05] calculated as the distance between q_k and x. In the following, we represent $\tilde{}(q_k,x) = (q_k,x)G(q_k,x)$. In order to accurately calculate the multiple scattering of light, a large number of sample points are required. However, this makes it difficult to calculate $L^n(x)$ in real-time. To address this problem, the proposed method precomputes the transmittance for the basic volume data, and the transmittance for the new volume data generated at run time is calculated using the precomputed transmittances. Moreover, the proposed method clusters the transmittances between each voxel and sample points (see Fig. 3(b)) in order to reduce the data size and accelerate the calculation of multiple scattering. The transmittances between each voxel and the sample points are clustered using a binary tree structure similar to [WFA*05] as shown in Fig. 3(c). Each leaf node corresponds to a sample point q_k and stores $\tilde{}(q_k,x)$. The interior nodes store the mean value $\langle\tilde{}\rangle$ and L^2 error due to clustering. A set of nodes whose errors are smaller than the user specified threshold represents the clusters of transmittances $\tilde{}$. This set of nodes is referred to as a cut, as shown in Fig. 3(c).

In the rendering process, the transmittance (q_k,x) for the density $(x) = w\ _i(x) + (1-w)\ _j(x)$ is calculated by:

$$\tilde{}(q_k,x) = G\exp(-\ _t\int_x^{q_k}(w\ _i(y) + (1-w)\ _j(y))dy).$$

Using the precomputed transmittances $\tilde{}_i(q_k,x)$ for $V_b(i)$ and $\tilde{}_j(q_k,x)$ for $V_b(j)$, $\tilde{}(q_k,x)$ can be calculated as:

$$\tilde{}(q_k,x) = \exp(w\log(\tilde{}_i(q_k,x)) + (1-w)\log(\tilde{}_j(q_k,x))). \tag{9}$$

Here, $\tilde{}_i$ and $\tilde{}_j$ are represented by cuts, and $\tilde{}$ is calculated by merging two cuts in a manner similar to [CPWAP08].

Next, the radiance of multiple scattered light is calculated iteratively at each voxel. First, the radiance $L^1(x)$ of single scattered light is calculated using Eq. (1) at each voxel, and the radiance of each sample point is interpolated using the radiances of the neighbor voxels. The radiance at each sample point is stored at each leaf node of the binary tree. The interior nodes are calculated by summing the radiances of two child nodes. This tree is referred to as the illumination tree. The radiance of the second scattered light at each voxel is calculated by summing the product of each node of the merged cut and the corresponding node of the illumination tree. The radiance of the second scattered light at each sample point is interpolated using the radiances of the neighbor voxels. By repeating this process, the radiance of multiple scattered light is calculated.

4. Results

Figure 4 shows still images of dynamic clouds rendered using the proposed method. Figure 4 shows the images of dynamic clouds in the daytime illuminated by blue sky and in

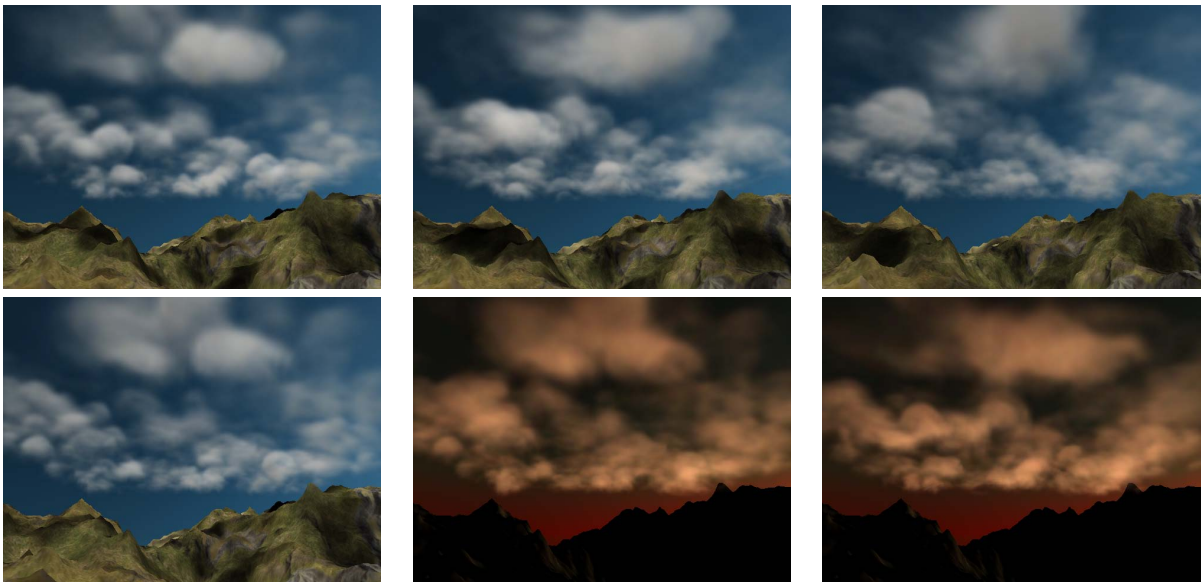

Figure 4: *Animation of dynamic clouds.*

the evening. Shadows of the clouds are rendered in the similar way to [DKY*00]. The number of basic volume data N_{vol} is 300 and the optical depth is represented by the 4-th order SH. The number of voxels N_{voxel} for the basic volume data is $128^2 \times 64$ and the number of sample points K is 2^{17}. The animation frame rate is about 26 fps on a standard PC with an Intel Core i7 X-980 CPU and NVIDIA GeForce GTX 580. The precomputational time of cuts for each basic volume is 12 min and that for SH coefficient vectors is 6 min. The resolutions of the images are 640×480.

5. Conclusion and Future Work

We have proposed an efficient method that realizes the real-time animation of dynamic clouds. The proposed method can create an endless animation of cloud using basic volume data. By computing the optical depths and transmittances for the basic volume data in advance, the proposed method can efficiently render dynamic clouds illuminated by sunlight and skylight with multiple scattering. We have implemented the proposed algorithms in CUDA, and by compressing the precomputed data to be fitted in GPU memory, real-time animation of dynamic clouds can be achieved.

In future work, we would like to develop an efficient calculation method for multiple anisotropic scattering effects.

References

[BNM*08] BOUTHORS A., NEYRET F., MAX N., BRUNETON E., CRASSIN C.: Interactive multiple anisotropic scattering in clouds. In *Proc. of Symposium on Interactive 3D graphics and games* (2008), pp. 173–182. 1

[CPWAP08] CHESLACK-POSTAVA E., WANG R., AKERLUND O., PELLACINI F.: Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Transactions on Graphics 27*, 5 (2008), Article No. 128. 3

[DKY*00] DOBASHI Y., KANEDA K., YAMASHITA H., OKITA T., NISHITA T.: A simple, efficient method for realistic animation of clouds. In *Proc. of SIGGRAPH 2000* (2000), pp. 19–28. 1, 4

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud rendering. *Computer Graphics Forum 20*, 3 (2001), 76–84. 1

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Transactions on Graphics 25*, 3 (2006), 977–986. 2, 3

[SKS02] SLOAN P. P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proc. of SIGGRAPH 2002* (2002), pp. 527–536. 2

[SKSU05] SZIRMAY-KALOS L., SBERT M., UMENHOFFER T.: Real-time multiple scattering in participating media with illumination networks. In *Proc. of Eurographics Symposium on Rendering* (2005), pp. 277–282. 1

[SSSE00] SCHODL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proc. of SIGGRAPH 2000* (2000), pp. 489–498. 1

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts : a scalable approach to illumination. *ACM Transactions on Graphics 24*, 3 (2005), 1098–1197. 3

[ZRL*08] ZHOU K., REN Z., LIN S., BAO H., GUO B., SHUM H.-Y.: Real-time smoke rendering using compensated ray marching. *ACM Transactions on Graphics 27*, 3 (2008), Article No. 36. 2