Fast and Realistic Display of Clouds Using a Recording Matrix

Y. Wu[1], B. Allgöwer[2], and D. Nüesch[1]

[1] Remote Sensing Laboratories, University of Zurich, Switzerland
[2]GIScience Center, University of Zurich, Switzerland

Abstract

For volumetric cloud models, traditional ray tracing methods and graphics hardware acceleration methods can result in photorealistic images, but with considerably expensive costs. Hence, real-time rendering can be achieved only under the condition of a static relationship between light sources and clouds. This paper presents a new method to accelerate the rendering process for animated clouds. In order to display visually convincing self-shading properties of clouds, multiple scattering among their internal particles is taken into account with the help of a recording matrix, which only requires small intermediate saving space. This method is not only flexible and easy to use, but also capable of rendering other dynamic gaseous phenomena such as smoke in a fast and efficient manner.

Keywords: multiple scattering, self-shading, recording matrix, scaling factor.

1. Introduction

Clouds play an important role in interactive flight simulations, games and weather forecasts. The inclusion of clouds in virtual scenes can enhance their visual reality. However, cloud rendering is difficult because realistic shading requires the consideration of complicated light transmission within cloud volumes.

Realistic cloud rendering should produce images of clouds as they are perceived by human eyes. Hence it can be seen as physically mimicking the process of light passing through clouds and the intermediate atmosphere before reaching a given view point. Because internal particles of clouds have high albedos [Boh87], multiple scattering of photons within clouds can not be ignored. Consequently, the light intensity at an internal position of clouds depends on the light arriving from source directly and after being scattered from other particles. A cloud image is formed when light scattered by the particles of the cloud reaches a given viewer after passing through the intermediate atmosphere. Accordingly, a two pass method has recently often been used to describe these physical processes [DKY*00, and HL01].

The two pass method leads to visually convincing cloud images no matter where a viewer stands. Its first pass is to calculate the light intensities on internal particles of clouds; and its second pass is to render all the particles by setting their colors to the products of light intensities and incoming light colors. The main advantage of this method is that the first pass only needs to be calculated once for a static cloud viewed from different directions unless its light sources change locations. But for dynamic clouds, internal particles changing positions continuously by wind, self evolution or other reasons, the first pass needs to be recalculated continuously after every internal movement. In addition, rapid movement of light sources in some games results in an obligatory recalculation of the first pass as well.

However, the first pass is a complicated and time-consuming step since multiple scattering among particles needs to be calculated. To reduce computation time, a new method is developed in this paper to process the first pass more efficiently. This method can render photorealistic cloud images with more than ten times faster speed as compared to the hardware acceleration method developed by Harris and Lastra [HL01].

2. Previous Work

Limited to the finite data processing speed nowadays, it is necessary to find an ideal compromise between cloud rendering speed and rendering effect. Accordingly, previous work on cloud rendering can be divided into two categories, mapping techniques and physical models, which stress rendering speed and rendering effect respectively. Mapping is a non-interactive rendering technique, which utilizes 2D or 3D textures with experimental data to form cloud images. Physical modeling is more desirable when visualizing clouds interactively, though it is more time-consuming.

Nishita et al. applied an anisotropic multiple scattering method to render clouds [NDN96]. Stam et al. simulated light scattering by blobs in the gases using ray tracing technique [FSJ01]. Dobashi et al. took advantage of graphics hardware acceleration technique and applied an approximation of isotropic single scattering to accelerate the rendering process [DNO98]. Hardware acceleration technique makes use of the computational capabilities of graphics hardware by applying texture mapping and color blending to compute the light intensities on metaballs caused by incident light and single forward scattering. Later on, still utilizing graphics hardware,

Harris and Lastra extended Dobashi's method with an approximation of multiple forward scattering and anisotropic first order scattering towards the viewer [HL01], which results in relatively brighter cloud images.

All the above methods obtained promising results with variable computation costs. Among them the hardware acceleration methods are most efficient, since for static clouds the light intensities on internal particles only need to be calculated once. Then real-time rendering can be achieved with the help of dynamic imposters [HL01]. However, for animated clouds, the recalculation of light intensities is unavoidable since it affects the self-shading properties of clouds. A crucial step in hardware acceleration technique is to read some pixels back from a color buffer for every particle. This is the bottleneck of speedup algorithm. Our method was therefore developed to overcome this shortcoming and achieve a higher frame rate.

This paper uses a metaball technique to represent internal particles of clouds. Metaballs are tiny spheres defined by centers, radiuses, and colors. They are replaced by billboards textured with a Gaussian function in rendering processes. Average light intensity on each metaball is saved to set its color. Further knowledge about metaballs and billboards can be found in [DKY*00, HL01]. To create visually convincing cumulus effects, we use a cube marching algorithm to construct cloud models.

3. Basic Illumination

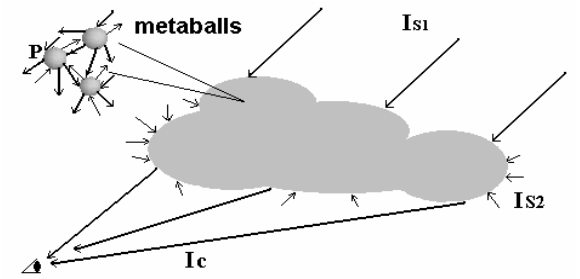

Figure 1: *Light transmission through a cloud and scattering towards a given view point.*

Fig.1 shows the basic processes of light transmitting through a cloud and subsequent scattering towards a viewer. Dominant sunlight and ambient light, with light intensities I_{S1} and I_{S2} respectively, are transmitted through the cloud and multiply scattered amongst the internal particles. Ambient light is composed of sky light and light reflected from other clouds as well as the ground, which should be taken into account according to the research of Nishita et al. [NDN96]. This process results in various light intensities on particles, which then scatter light through the cloud and intermediate atmosphere to the viewer with intensity I_C .

Accordingly, the first pass involves the calculation of the light intensities on metaballs, which determine the self-shading details of clouds. The average intensity I_p on a metaball p is composed of the light from light sources which not absorbed by intervenient metaballs, and light scattered on it by other metaballs. Accordingly, we approximate I_p with the following discrete equation:

$$I_P = \sum_{n=1,N} I_{sn}(l_n)\cdot(1-\prod_{i=1}^{M1} e_i) + \sum_{x\in A} B(x)\cdot(1-\prod_{i=1}^{M2} e_i) \quad (1)$$

where l_n is the direction from the n[th] light source to the metaball p , $I_{sn}(l_n)$ is the intensity of the n[th] light source, e_i is the extinction ratio of intervenient metaball i through which light casts to p , $M1$ and $M2$ are the numbers of intervenient metaballs, A represents the aggregation of metaballs which scatter light to p, and

$$B(x) = \sum_{w\in C} a(x)\cdot I(x,w)\cdot r(w,w') \quad (2)$$

is the intensity of light scattered from metaball x towards p. In Eq. 2, C represents the directions of all the incident light at x , $a(x)$ is the albedo of the metaball at x , $I(x,w)$ is the intensity of the incident light from direction w , w' is the direction from x to p, and $r(w, w')$ is the phase function, which determines the percentage of light scattering to w' from incident light in direction w .

It is complicated to fully calculate multiple scattering inside a cloud according to Eq. 1 and 2 directly, because each cloud is composed of many metaballs and light is repeatedly scattered among them back and forth. Fortunately, the work of Nishita et al. demonstrated that the contribution of multiple scattering is dominated by the first two orders [NDN96]. Hu et al. also found that cloud phase function often generates high values along forward directions within a narrow solid angle less than 15 degrees, and then values sharply decrease according to the angle between incident and scattered light [HBB*00]. Therefore, in order to accelerate the illumination process, we model the multiple scattering of dominant light as multiple forward scattering and ambient light as multiple supplementary forward scattering. Here we must notice that ambient light is a general name for light having much weaker intensity than dominant light. When an ambient light source becomes comparatively strong, it must be considered as a dominant light source.

To further simplify computation, we assume the varying albedo $a(x)$ to be a constant, α . The extinction ratio e_i is also assumed to be a constant, e for all metaballs. Moreover, since forward scattering occurs within a narrow solid angle, the phase function is assumed to be a constant r too. For a single light source, Eq.1 is simplified to:

$$I_p = I_{p-1}\cdot(1-e) + I_k\cdot\alpha\cdot r \quad (3)$$

where I_{P-1} is the average intensity at places through which the incident light directly arrives p , I_k is the average intensity at places from which scattered light directly cast on p .

Ambient light is taken into account as a supplement to prevent over-extinction of dominant light. It comes from many sources and hence is hard to simulate exactly. We simply assume that ambient light has isotropic distribution surrounding a cloud, and its effect on the same side of the dominant light source has been included as part of dominant

light. Since dominant light is most attenuated along its forward direction, the contribution of ambient light along the inverse direction is the most marked and worthy of taking into account.

To prevent overlightening a metaball, we simply assume that a light source with weaker intensity can not enhance the intensity on an object having stronger intensity. This assumption prevents the colors of metaballs all turning into full white and hence eliminating the self-shading properties of clouds. Therefore, the supplementary intensity to a metaball is calculated by multiple forward scattering the differences between the intensity of ambient light and the intensities already on metaballs caused by dominant light. By similar reasoning, if there is another light source, we only need to calculate its multiple supplementary forward scattering to constrain the brightness of clouds.

The second pass computes the light intensities scattered by metaballs towards the viewer. If we ignore the scattering and extinction of intermediate atmosphere, the intensity E_p scattered by metaball p to the viewer is:

$$E_p = E_{p-1} \cdot (1 - e_i) + I_p \cdot \alpha \cdot r(\theta) \qquad (4)$$

where E_{p-1} is the intensity scattered by other metaballs through p to the viewer, and $r(\theta)$ is Simple Rayleigh phase function [HL01].

4. Rendering Method

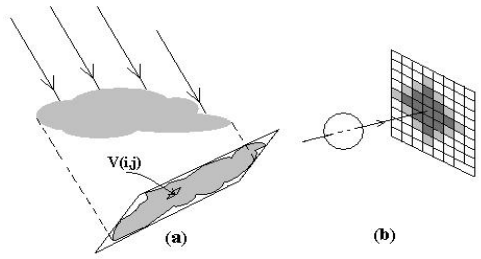

Figure 2: *Recording matrix and related elements for calculating the average intensity on a metaball.*

To calculate the average light intensity on a metaball according to Eq.3, in this paper a matrix is exploited to record each light intensity change along ray clusters. A matrix is a small array in CPU. To be understood more easily, it can be viewed as an image perpendicular to incident light and fully containing the projection of a cloud, as shown in Fig.2 (a). Each pixel in this image stores a float value which is an element $V(i, j)$ of the recording matrix. To avoid redundant computation, we set the resolution of the recording matrix to be the same as user's computer screen. Light intensities on metaballs then can be calculated by repetitively reading and revising the stored intensities from the matrix. It is important to mention that there is no need to store intensities in red, green, and blue respectively, because they change in a uniform ratio. A colorful cloud can be generated by multiplying the color of incident light with the calculated intensities on metaballs.

In implementation, all elements in the recording matrix are first set to the intensity of incident light. Then the computation is performed on metaballs of increasing distance towards a light source, according to the forward scattering assumption. To calculate the intensity on a metaball p, the first step is to project p onto the recording matrix shown in Fig. 2(b). Its projection occupies the dark gray elements whose average is I_{P-1} in Eq. 3. Its surrounding light grey elements are simply assumed to represent the places which can also scatter light on p. Hence, I_k is the average of their values and I_{P-1}. Then I_P is computed and set to the dark grey elements to record new light intensities after p.

After the calculations have been performed for all metaballs, the matrix finally records the intensities on the metaballs that lie on the far side of light source. Since we assume that ambient light only transmits in places that have lower intensities along the backward direction of dominant light, the contribution of ambient light is calculated by multiple forward scattering of the differences between the intensities of ambient light and the values in recording matrix. Hence, the matrix is updated to store the differences and multiple forward scattering is performed again in inverse order. The final intensity on a metaball is the sum of the intensities resulting from dominant light and ambient light.

To further accelerate the first pass, a selectable scaling factor is used to shrink the cloud model. This factor is flexible and can be selected depending on metaball size and the resolution required by users. Our experiences found that for 20000 metaballs having radiuses of around 10 pixels on computer screen, a scaling factor of 0.5 causes a hardly distinguishable difference in the final image, but with a speed increase of five times as compared to the computation without the scaling factor.

In the second pass, hardware blending is used to render the final image according to Eq. 4. The metaballs are sorted again based on their distances to the viewer and rendering is performed in order of descending distance.

5. Results

Our method can quickly display photorealistic clouds with small storage costs. Fig. 3 displays a close-up view of a cloud constructed by 24000 metaballs; rendering took 0.062 seconds. Fig.4 shows clouds constructed by 36000 metaballs colored by sunset; rendering took 0.078 seconds in this case. Fig. 5 shows a cloud also constructed by 24000 metaballs under weak light and heavy extinction. The above figures were all rendered with a scaling factor 0.5. The left cloud in Fig. 6 was also rendered with a scaling factor 0.5; rendering cost 0.07 seconds. The middle cloud in Fig. 6 shows the cloud rendered without a scaling factor; the cost was 0.17 seconds. The right cloud in Fig. 6 shows the cloud rendered by the hardware acceleration method [HL01] without taking ambient light into account. Rendering took 0.78 seconds. Fig. 7 displays the time costs for the first pass using our matrix method and the hardware acceleration method [HL01]. All images and data were obtained on a PC with Pentium4 2.4GHz and NVIDIA GeForce 5700LE graphics processor.

Figure 3: *A close-up view of a cloud.*

Figure 4: *Clouds colored by sunset.*

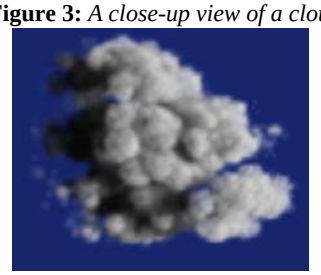

Figure 5*: A cloud under weak light.*

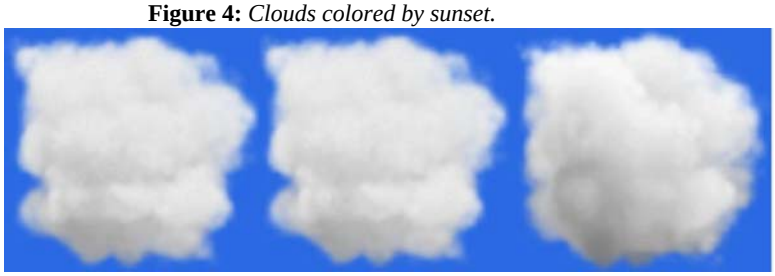

Figure 6*: A cloud rendered by our method and hardware acceleration method.*

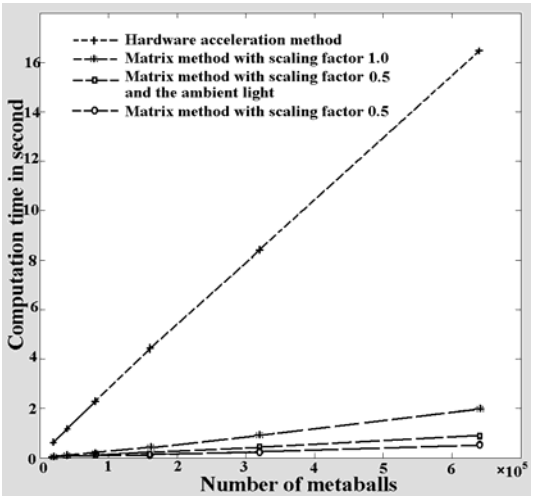

Figure 7: *Computation costs.*

6. Conclusion

This paper presented a fast and efficient cloud rendering method for both static and animated clouds. This method reads and revises intermediate illumination results into a recording matrix, which is a small array related to the resolution of user's computer screen. A flexible scaling factor further accelerates rendering without significant loss of self-shading details. Our method is also useful for the efficient rendering of other gaseous phenomena, such as smoke.

Acknowledgements

This work is funded by the Swiss National Science Foundation within the National Research Program 48 'Landscapes and Habitats of the Alps'.

References

[Boh87] Bohren C.F.: Multiple Scattering of Light and Some of its Observable consequences. *Am. J. Phys. , 1987*, Vol.55 (6), pp.524-533.

[DKY*00] Dobashi Y.,Kaneda K., Yamashita H., Okita T., and Nishita T.: A Simple, Efficient Method for Realistic Animation of Clouds. In *Proc. SIGGRAPH'00,* pp.19-28.

[DNO98] Dobashi Y., Nishita T., Okita T.: Animation of Clouds Using Cellular Automaton. In *Proc. Computer Graphics and Imaging'98*, pp. 251-256.

[FSJ01] Fedkiw R., Stam J., Jensen H.W.: Visual Simulation of Smoke. In *Proc. SIGGRAPH'01*,pp. 15-22.

[HBB*00] Hu Y.X., Wielicki B., Lin B., Gibson G., Tsay S.C., Stamnes K., Wang T.: δ-Fit: A Fast and Accurate Treatment of Particle Scattering Phase Functions with Weighted Singular-value Decomposition Least-squares Fitting. *Journal of Quantitative Spectroscopy & Radiative Transfer 65, 2000*, pp. 681-690.

[HL01] Harris J.M., Lastra A.: Real-Time Cloud Rendering. In *Proc. Eurographics2001*, Vol. 20(3), pp.76-84.

[NDN96] Nishita T., Dobashi Y., Nakamae E.: Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light. In *Proc. SIGGRAPH'96*, pp.379-381.