Light Propagation for Mixed Polygonal and Volumetric Data

Caixia Zhang*, Daqing Xue*, Roger Crawfis*
The Ohio State University

ABSTRACT

Some applications require scenes mixing polygonal and volumetric objects and shadows make the scenes more realistic. This paper describes a shadow algorithm for mixed polygonal and volumetric data, including the generation of soft shadows for area light sources. Our volume shader leverages advanced graphics GPU for an accelerated and feasible solution. The shadow and soft shadow algorithm applies to all combinations of volumes and polygons, without any restriction on the geometric positioning and overlap of the volumes and polygons.

For realistic rendering where we have a high albedo participating media, multiple scattering is significant. We extend our algorithm to handle both multiple forward scattering and back scattering with light attenuation. This constitutes a complete system for shadow generation and light propagation.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Display Algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Color, shading, shadowing and texture.

Keywords: visualization, shadows, soft shadows, multiple scattering, illumination

1 INTRODUCTION

A shadow is a region of relative darkness within an illuminated region caused by an object totally or partially occluding a light source. Shadows are essential to realistic images. In volume rendering, as the light traverses the volume, the light intensity is continuously attenuated by the volumetric densities. The generation of soft shadows for extended light sources is a difficult topic in computer graphics. It requires integrating the contributions of extended light sources for each point of illumination on an object.

Some applications require both volumetric and geometrical objects together in a single image. For example, geometrically defined objects may be surrounded by clouds, smoke, fog, or other gaseous phenomena. In this paper, we will generate shadows and soft shadows for a scene including both volumetric datasets and polygonal geometries using a texture-based volume rendering method.

For a high-albedo participating media, such as clouds, multiple scattering cannot be ignored. Here, we implement multiple forward scattering and back scattering, and incorporate the multiple scattering into our shadow algorithm.

In the following section, previous work is reviewed and motivation for this work is given. Section 3 discusses multiple scattering for high-albedo participating media. Section 4 describes our shadow algorithm for mixed polygonal and volumetric data, while Section 5 describes the soft shadow algorithm for a scene including both volumes and polygons. Conclusions and future work are given in Section 6.

*{zhangc, xue, crawfis}@cse.ohio-state.edu

2 PREVIOUS WORK

Our work is related to both light transport and the rendering of mixed polygonal and volumetric data. In this section, we will briefly summarize prior research in shadow algorithms, multiple scattering and rendering techniques for mixed volumes and polygons.

2.1 Shadow Algorithms

Earlier implementations of shadows focused primarily on hard shadows, in which a value of 0 or 1 is multiplied with the incoming light intensity. The shadow volume algorithm by Crow [3] introduces the concept of shadow volumes. A 2-pass hidden surface algorithm is proposed by Nishita and Nakamae [21,22] and Atherton et al. [1]. Williams [30] uses a z-buffer depth-map algorithm to generate shadows.

These shadow algorithms can only determine if a point on an object is in shadow or not, resulting in only binary values for the light intensity. These algorithms are not suitable for volume rendering. In volume rendering, as the light traverses through the volume, the light intensity is continuously scattered and attenuated by the volumetric material. Ray tracing offers the flexibility to deal with the attenuation of the light intensity and has been used to generate shadows for both surface representations [29] and volumetric datasets [4,9]. Lokovic and Veach [16] proposed the concept of deep shadow map to keep track of the light attenuation in the volume. For splatting, Nulkar and Mueller[24] have implemented an algorithm to add shadows to volumetric scenes using a 3D buffer. A new algorithm has been proposed to implement shadows and soft shadows using splatting that requires only a 2D buffer for each light source [32,33]. Kniss et al. [11,13] also utilize an off screen buffer to accumulate the light attenuation in their volume rendering using 3D texture slicing.

2.2 Multiple Scattering

For low albedo media, the scattering is insignificant compared to the light attenuation. However, for realistic rendering of high albedo participating media, for example, clouds or water vapor, multiple scattering is very important. Multiple scattering must account for scattering in all directions. It is more physically accurate, but much more complicated and expensive to evaluate. Max [18] gives an excellent survey of optical models, including multiple scattering.

The calculation of multiple scattering can be divided into four methods [18]: the zonal method, the Monte Carlo method, the P-N method and the discrete ordinates method. In the zonal method [26], the volume is divided into a number of finite elements which are assumed to have constant radiosity. This method is valid only for isotropic scattering. In the Monte Carlo method [27], a random collection of photons or flux packets are traced through the

volume, undergoing random scattering and absorption. The resulting images tend to appear noisy and/or take a long time to compute. The P-N method [2,9] uses spherical harmonics to expand the light intensity at each point as a function of direction. The discrete ordinates method uses a collection of M discrete directions, chosen to give optimal Gaussian quadrature in the integrals over a solid angle. Lathrop [14] points out that this process produces ray effects and presents modifications to avoid these ray effects. Max [19] describes an approximation to the discrete ordinates method, which reduces the ray effects by shooting radiosity into the whole solid angle bin, instead of in a discrete representative direction.

Recently, some research on approximate methods to multiple scattering has been examined to achieve real-time rendering. Harris and Lastra [7] provide a cloud shading algorithm that approximates multiple forward scattering along the light direction. Kniss et al. [12,13] use an empirical volume shading model and add a blurred indirect light contribution at each sample. They approximate the diffusion by convolving several random sampling points and use graphics hardware to do the volume rendering. All their work used slice-based methods. Some other work [4,5,20] also used slice-based techniques to calculate the scattering effects in volume.

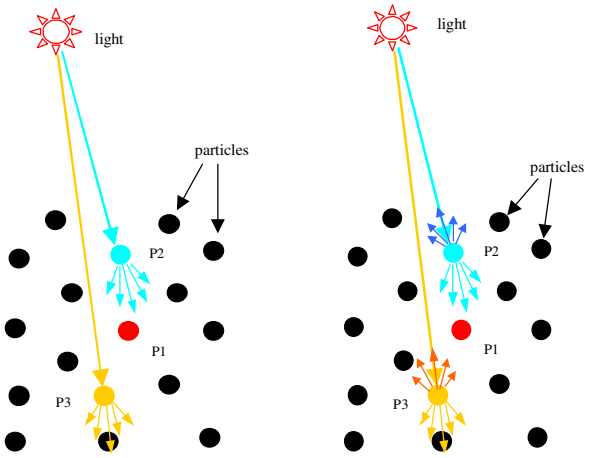
2.3 Rendering both Volumes and Polygons

To render scenes mixing volumetric and polygonal models, the most common solution is to convert the polygonal and volumetric data into a common representation: either construct surface polygons from volume data [17] or change polygon data to volume data using 3D scan-conversion [10]. This conversion introduces artifacts and is generally expensive and inefficient. An alternative approach is to directly render both data types. Levoy has developed a hybrid ray tracer for rendering polygon and volume data [15]. Rays are simultaneously cast through a set of polygons and a volume data array, samples of each are drawn at equally spaced intervals along the rays, and the resulting colors and opacities are composed together in a depth-sorted order. In Levoy's method, both volume and polygon objects are rendered using ray tracing. Ebert and Parent use another method which combines volume rendering and scanline a-buffer technique [6]. The scanline a-buffer technique is used to render objects described by surface geometries, while volume modeled objects are volume rendered. The algorithm first creates the a-buffer for a scanline, which contains a list of all the fragments of polygons for each pixel that partially or fully cover that pixel. Based on the scanline-rendered a-buffer fragments, the volume-modeled objects are broken into sections and combined with the surface-defined a-buffer fragments. In their paper, the volumes are defined by procedural functions to model gaseous phenomena.

Our motivation is to implement multiple forward and backward scattering efficiently, and generate shadows and soft shadows for scenes of mixed polygonal and volumetric data. Our focus is on efficient volume rendering schemes such as 3D texture mapped approach.

3 Multiple Scattering

In order to simulate light transport for participating media with high albedo, multiple scattering cannot be ignored. In this section, we will explain how we implement multiple scattering using a texture-based rendering method and incorporate it with the shadow algorithm by displaying clouds, a high albedo participating medium. The clouds are modeled as a collection of ellipsoids, and Perlin's fractal function [25] is used to disturb the density distribution. Figure 2 shows the clouds with the light emanating behind it. The clouds look unnaturally dark, because



(a) (b)

Figure 1: A schematic of light transport (a: forward scattering; b: forward scattering and back scattering)

only light attenuation is modeled. In this section, we will explain our implementation of multiple scattering and its effects on the cloud appearance.

For multiple scattering, the light intensity at a point ***P*** is the sum of the direct energy from the light source that is not absorbed by intervening particles and the energy scattered to ***P*** from all other particles. Figure 1 is a schematic showing forward scattering and back scattering among particles. The calculation of multiple scattering requires accounting for the scattering from all directions. We let $I(P,\omega)$ be the intensity at each point P and each light flow direction ω, which can be expressed as:

$$I(P,\omega) = I_0(\omega)\cdot e^{-\int_0^D \tau(P-t\omega)dt} + \int_0^D g(s,\omega)\cdot e^{-\int_0^s \tau(P-t\omega)dt} ds \quad (1)$$

where $I_0(\omega)$ is the original light intensity in direction ω, τ is the extinction coefficient of the participating media, D is the depth of ***P*** in the media along the light direction, and

$$g(s,\omega) = \int_{4\pi} r(P-s\omega,\omega,\omega')\cdot I(P-s\omega,\omega')d\omega' \quad (2)$$

represents the light from all directions ω' scattered into direction ω at the point ***P***. If we denote $P-s\omega$ as ***x***, then $r(x,\omega,\omega')$ is the bi-directional scattering distribution function (BSDF) which determines the percentage of light incident on ***x*** from direction ω' that is scattered in direction ω. We can treat $r(x,\omega,\omega') = a(x)\cdot\tau(x)\cdot p(\omega,\omega')$ [7, 18], where $a(x)$ is the albedo of the media at ***x***, $\tau(x)$ is the extinction coefficient of the media at ***x***, and $p(\omega,\omega')$ is the phase function.

A full multiple scattering algorithm must compute this quantity for all light flow directions. This would be very expensive. Nishita et al. [23] take advantage of the strong forward scattering characteristics of many volumetric materials and limit the sampled light flow directions to sub-spaces of high contribution. Harris et al. [7] further approximate multiple forward scattering only in the light direction, resulting in good cloud images. However, they did not model back scattering. As pointed out by Max [18], without back scattering, the edges of very dense clouds are not illuminated properly. In this paper, we model both multiple forward scattering as well as back scattering and we approximate the multiple scattering along the light direction based on the strong forward

Figure 2: Clouds without multiple scattering

Figure 3: Clouds with multiple forward scattering only

Figure 4: Clouds with both multiple forward scattering and multiple back scattering

scattering characteristics of clouds for the purpose of efficient calculation.

We approximate the integration of (2) over a solid angle γ around the light direction. Here, ω' is within the solid angle γ along the light direction $\boldsymbol{l}$.

$$g_k^f = \int_{\gamma_A} a_k(x) \cdot \tau_k(x) \cdot p(l, l_{\gamma_A}) \cdot I(x, l_{\gamma_A}) dl_{\gamma_A} \quad (3)$$

$$g_k^b = \int_{\gamma_B} a_k(x) \cdot \tau_k(x) \cdot p(l, l_{\gamma_B}) \cdot I(x, l_{\gamma_B}) dl_{\gamma_B} \quad (4)$$

The above g_k^f represents the forward scattering and g_k^b is for the back scattering. γ_A and γ_B are solid angles for the forward scattering and the back scattering respectively. l_{γ_A} and l_{γ_B} are directions within γ_A and γ_B. The subscript $\boldsymbol{k}$ here indicates that the values are from the slice $\boldsymbol{k}$.

Now, assume we have accurately calculated the flux for each point in a plane perpendicular to the light direction. Let $I(x,l)$ be the light distribution in a slice, the above formulas for g_k^f and g_k^b can be calculated using a convolution over the light intensity $I(x,l)$. The convolution calculation is supported and easy to implement in texture-based rendering. This shows that multiple scattering can be modeled as light diffusion along the light direction. Kniss et al. [12,13] also use a convolution technique to model the light diffusion, but they only model multiple forward scattering.

If only multiple forward scattering is considered, we have the recurrence relation:

$$I_k = \begin{cases} g_{k-1} + T_{k-1} \cdot I_{k-1}, & 2 \le k \le N \\ I_0, & k = 1 \end{cases} \quad (5)$$

where, $T_{k-1} = e^{-\tau_{k-1}}$ is the transparency of slice ***k-1***. The recurrence relation says the light incident on slice ***k*** is equal to the intensity scattered to slice ***k*** from the slice ***k-1*** plus the direct light transmitted to the slice ***k***. Here, g_{k-1} is calculated using a convolution operation based on (3).

For multiple forward scattering, we just need to keep the intensity of the previous slice, then calculate the multiple forward scattering g_{k-1} using convolution over I_{k-1} and add it to I_k. Figure 3 shows the clouds with multiple forward scattering only. The clouds look brighter than the clouds in Figure 2 without multiple scattering.

From the physical viewpoint, when the light is scattered forward, some light will scatter backwards. In Figure 1(a), P1 receives energy from P2 by forward scattering, while in Figure 1(b), P1 also gets energy from P3 by back scattering. In order to implement back scattering, we keep the light intensity of the slices which contribute to the current slice ***k***. The light intensity stored is the sum of the direct light plus the multiple forward scattering. And the back scattering is calculated using a convolution operation based on (4). Once a slice obtains new energy from back scattering, it will add this to the stored light intensity, and then bounce back energy to its upper slice. This process continues until the slice ***k*** to be illuminated is reached. After the illumination, the total energy at the slice ***k*** is scattered forward to slice ***k+1***. For the practical implementation, we maintain a limited number of slices, for example, three slices, and calculate their back scattering contribution on the current slice.

If we add back scattering to the clouds in Figure 3, we get the clouds in Figure 4. As pointed out by Max [18], as energy bounces around, the edges are brighter, as is the interior region of the clouds.

4 Shadow Algorithm Combining Volumes And Polygons

Based on the shadow algorithm for volumetric data [32], we can also generate shadows for scenes of mixed polygonal and volumetric data. We assume the polygons are opaque in this section to aid in our description of the algorithm.

4.1 Shadow Algorithm

Our shadow algorithm for mixed polygonal and volumetric data has two stages. First, the polygons are partially rendered with respect to both the viewer and the light source. For the second stage, texture-based rendering is used to render the volumes, and any relevant polygon information is composited into the scene slice by slice with proper shadow attenuations. The depth information from the rendering of opaque polygons is combined with the shadow buffer which stores the accumulated light attenuation to determine the shadow value. Illumination of both polygons and volumes is thus calculated during this second stage. Next we will explain our algorithm in detail.

The first stage is to render the polygons. Since our purpose is to generate shadows, the polygons are rendered with respect to both the viewer and the light source. At this stage, the light attenuation is not determined for the polygons, so no illumination is calculated and the intermediate rendering results are stored for

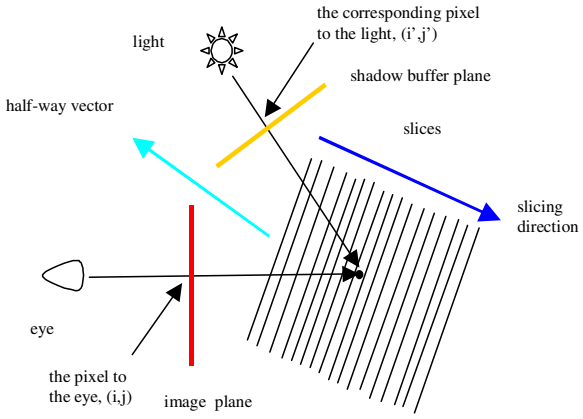


Figure 5: A schematic of the slicing direction and the buffers

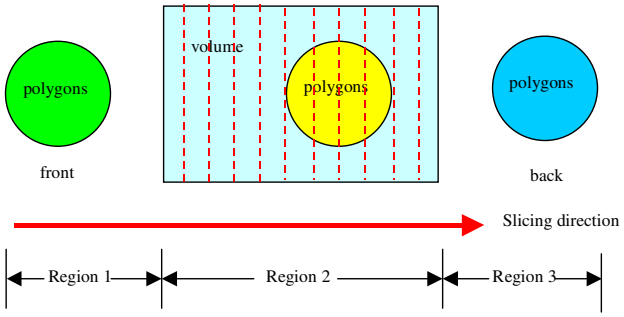


Figure 6: Position relationship between polygons and volume with respect to the slicing direction

final rendering. For the image generation, we store the z value with respect to the viewer into the z-buffer, and store the object color and normal into two textures. For the shadow generation, we save the z value with respect to the light into another texture.

The second stage is to render the volumes and composite the polygon information slice by slice. Similar to the shadow algorithm for volumes only [32], the volume is sliced along the half-way vector between the eye vector and the light vector, in order to keep track of accurate light attenuation information. The image buffer is aligned with the eye, and the shadow buffer is aligned with the light source (as shown in Figure 5).

Polygons in a volumetric scene can be surrounded by transparent air, or inside a semi-transparent volumetric participating media. If there are several volumetric datasets in a scene, we treat them as a single large volume. The positioning relationship between the polygons and the non-empty volume is shown in Figure 6, with respect to the slicing direction. Region 1 can cast shadows through the volume, as well as on objects within region1. In region 2, volumetric shadows are needed as well as volumetric shadowing of interior polygons. For region 3, shadows are cast from the prior regions.

The polygons at different regions are processed in different ways. All the polygons in region 1 are rendered in one step. Whether a point is in shadow or not is determined by the depth value with respect to the light stored in the texture. After rendering the polygons in this region, the shadow of the polygons is used for the first volumetric slice.

The region 2 is rendered slice by slice. The contribution of the volume data at the current slice is composited into the frame buffer with the z-test enabled so that only the volume in front of the polygons is rendered. The polygons in region 2 are rendered

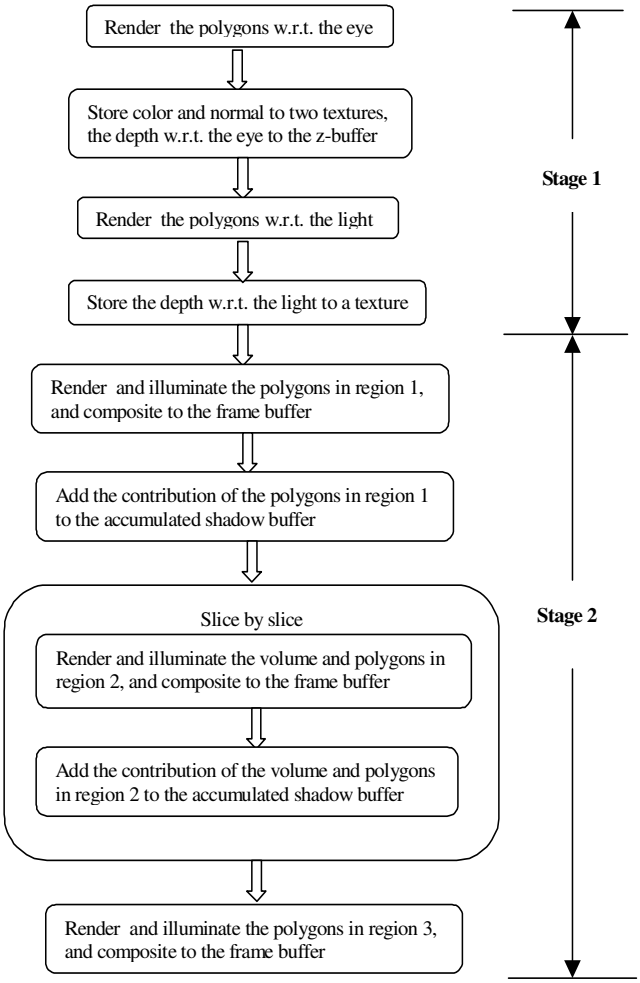
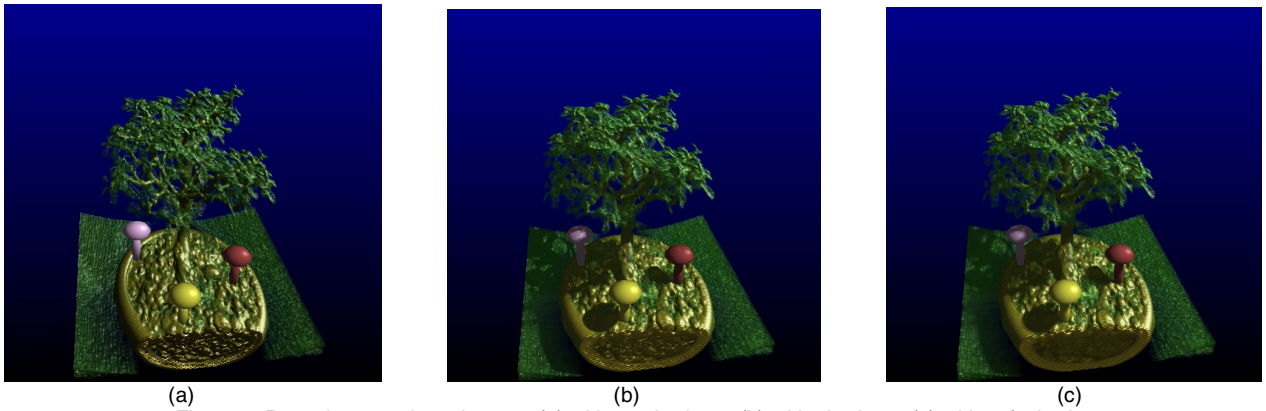


Figure 7. Flow chart of the shadow algorithm combining volumes and polygons

slice by slice, because the light attenuation by the volumetric data changes slice by slice. Whether there are polygons in the current slice is determined by the depth information of the polygons. If there are polygons at the current slice, the part of the polygons in this slice is rendered. The contribution of the polygons in the current slice is composited into the frame buffer with the opacity set to 1.0 at the corresponding pixels. Similarly, the contributions of the volumes and the polygons in this slice on the light attenuation are composited to the accumulated shadow buffer aligned to the light source for the illumination of the next slice. In region 2, the shadow value of a pixel is determined by a corresponding pixel on the accumulated shadow buffer (as shown in Figure 5).

The polygons in region 3 are rendered in one step in a similar way to the polygons in region 1, but they are different in how they are shadowed. We cannot just use the shadow z-buffer to determine the shadow values for the polygons in region 3, since we need to consider the shadows cast by the volume in region 2. So, we use both the accumulated shadow buffer and the z-buffer to determine the shadow values for the polygons in this region. The opacity from the z-buffer is set to 1 or 0 depending on whether there is a polygon occluder. We then take the maximum opacity value as the shadow value for the pixels corresponding to the polygons in region 3.



(a) (b) (c)

Figure 8: Bonsai tree and mushrooms (a) without shadows, (b) with shadows, (c) with soft shadows

Figure 9: Bonsai tree and mushrooms

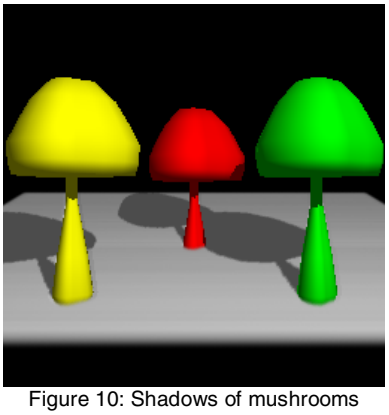


Figure 10: Shadows of mushrooms

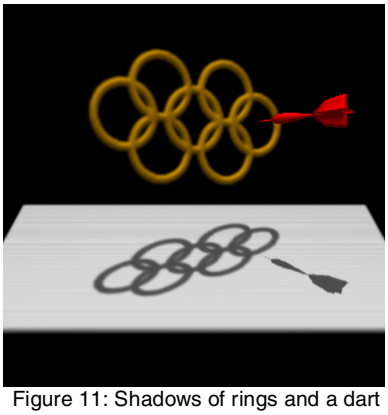


Figure 11: Shadows of rings and a dart

Figure 12: A scene of a teapot inside a translucent cube

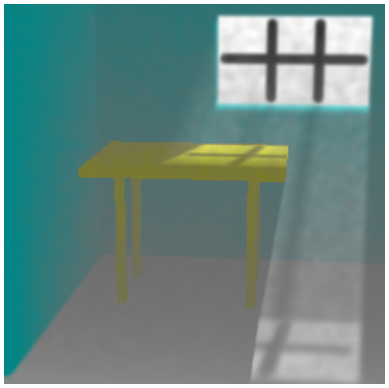


Figure 13: A scene of a desk inside a smoky room

Figure 14: An airplane flying above the clouds

The above shadow algorithm for mixed volumes and polygons applies to all the possible configurations of volumes and polygons, without any restriction on the geometric positioning and overlap of the volumes and polygons. For special cases, pure volumes are rendered in region 2, and pure polygons are rendered in region 1.

The region division aids in the efficiency of the algorithm. The polygons within only air can use a slice skipping technique and are rendered faster. We summarize the shadow calculation of the three regions in table 1.

The shadow algorithm combining volumes and polygons using texture-based rendering is demonstrated with the flow chart in Figure 7.

Table 1. Shadow determination of three regions

Region 1	Region 2	Region 3
Shadow z-buffer	Shadow z-buffer + Volume slices	Projective shadows + Shadow z-buffer

In cases of semi-transparent polygonal objects, we cannot render the polygons in one step, because the light attenuation is accumulated slice by slice. Also, we cannot set the opacity to 1.0 and use the depth stored in z-buffer to do z-test for the rendering of the volumetric data. So, the first stage, which renders the polygons first and stores the depth information with respect to the light into a texture and the depth with respect to the eye into the z-

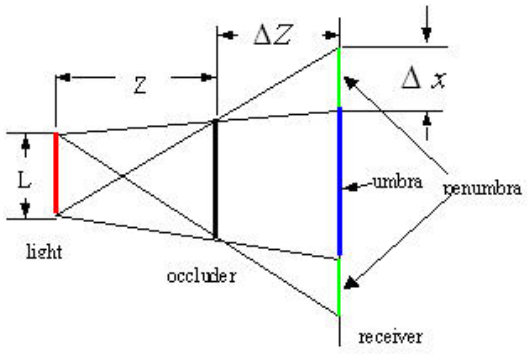


Figure 15: A schematic of the light source, occluder, and the receiver

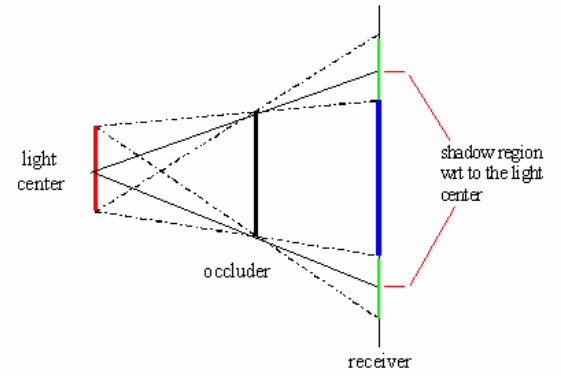


Figure 16: A schematic of the shadow region with respect to the light source

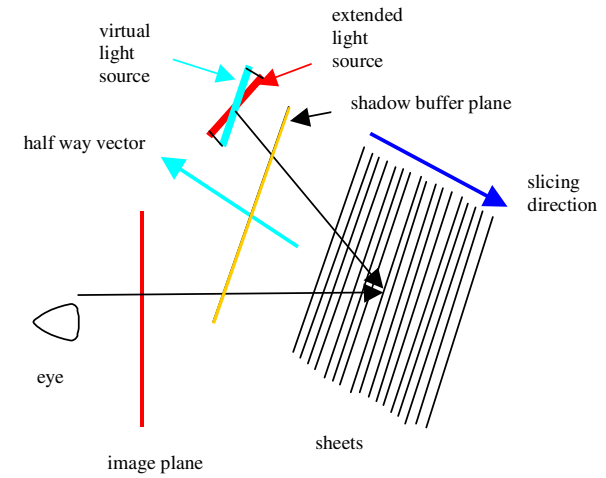


Figure 17: A schematic of the slicing direction and the buffers for the soft shadow algorithm

buffer, is unnecessary. Both the polygonal and volumetric objects are rendered slice by slice, with respect to both the eye and the light source, respectively. This is actually the case of pure region 2.

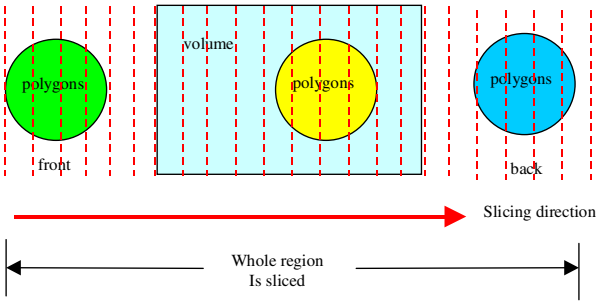


Figure 18: The whole region rendered slice by slice

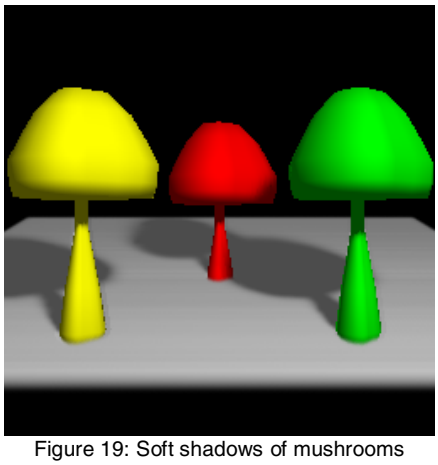


Figure 19: Soft shadows of mushrooms

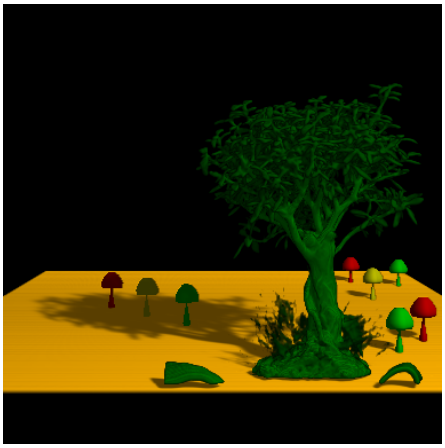


Figure 20: Soft shadows of Bonsai tree and mushrooms

4.2 Shadow Results

Using the above shadow algorithm, we have implemented shadows for scenes including both volumes and polygons. For Figures 8 – 14, please also see the color plate.

Figure 8 and Figure 9 show some polygonal mushrooms under the volumetric Bonsai tree. Figure 8(a) is the image without shadows and Figure 8(b) is with the shadows. The tree casts shadows on the bottom plate and the container as well as on the mushrooms. Compared with the image without shadows in Figure 8(a), the shadows in Figure 8(b) provide the spatial relationship and make the scene more informative and realistic. In Figure 9, some mushrooms cast shadows on the bottom plate and the other three mushrooms are in the shadows of the Bonsai tree.

Figure 10 shows the shadows cast from three mushrooms. The mushrooms are represented as polygons, while the bottom plate is represented as volumetric data. We can see the polygonal mushrooms cast shadows on themselves and on the plate. Figure 11 shows a scene in which a dart is pointing at the rings. Here, the dart is a polygon model, and the rings and the plate are volumetric data.

Figure 12 and Figure 13 are two scenes in which polygons are inside semi-transparent volumetric data. Figure 12 shows a polygon teapot inside volumetric translucent media. We can see the shadows cast by the teapot on itself and through the translucent media. In Figure 13, light comes into the room from the back and a polygon-defined desk resides in the smoky room. Here, the smoke is modeled using Perlin's turbulence function [25].

Our shadow algorithm for combining volumes and polygons also works for high-albedo media. In Figure 14, a polygonal airplane flies above clouds. The clouds are modeled with light attenuation and multiple scattering.

5 Soft Shadow Algorithm Combining Volumes And Polygons

The generation of soft shadows requires integrating the contributions of extended light sources on the illumination of objects. Soler and Sillion [28] proposed a soft shadow algorithm using a convolution technique. A soft shadow algorithm for volumetric data using sheet-based splatting was implemented in [33]. The soft shadow algorithm is an analytic algorithm using convolution techniques. If the light source is parallel to the slices, the width of the penumbra region (as shown in Figure 15) is calculated using the formula:

$$\Delta x = \frac{L \times \Delta Z}{Z} \tag{6}$$

The soft shadows are generated by convolving the shadows with respect to the center of the extended light source (in Figure 16), which is generated using the shadow algorithm discussed in Section 4. In the soft shadow algorithm, the volume is still sliced along the half-way vector between the eye vector and the light vector (as shown in Figure 17). A virtual light source parallel to the slices is constructed and the shadow buffer is parallel to these slices (in Figure 17). The soft shadow algorithm is evaluated and possible artifacts are discussed for more general cases in [33].

When there are polygons in the scene, we will render the polygons slice by slice in the same way as the volume, no matter where the polygons are (as shown in Figure 18). This is because the accumulated shadow buffer is convolved slice by slice, even for the pure polygons. So, we treat both the volumes and the polygons as a whole volume.

Similar to the shadow algorithm in Section 4, there are two stages for the soft shadow algorithm. The first stage is mainly to render the polygons with respect to the viewer and store the depth information of the polygons into the z-buffer.

At the second stage, texture-based rendering is used to render the volumes and polygons, and generate soft shadows slice by slice. In order to generate soft shadows for polygons, the polygons are not rendered in one step as in Section 4. Instead, they are rendered slice by slice using two clipping planes, regardless as to whether they are inside the volumetric material or not. Therefore, the range of the volume to be rendered slice by slice is determined by both volumes and polygons.

At each slice, the volumes in the current slice are rendered with the z-test enabled, and the part of the polygons belonging to the current slice is then rendered. Their contribution to the light attenuation is used to update the shadow buffer. Then the updated accumulated shadow buffer is convolved to prepare for the next slice. Since the polygonal objects are modeled as the boundary polygons of the solid objects, we should use the stencil buffer to add the contribution of the solid polygonal objects to the shadow buffer for the convolution. In this way, we get realistic soft shadows.

For semi-transparent polygonal objects, the first stage is unnecessary, because the main function of the first stage is to keep the depth information with respect to the eye into the z-buffer for the z-test of the rendering of the volumetric data at the second stage. Now the rendering of the volumetric data cannot use z-test to do the z-culling.

We have generated soft shadows for scenes with both volumes and polygons, using the above soft shadow algorithm. For Figures 19 and 20, please also see the color plate.

Figure 19 shows the soft shadows of the mushrooms. We can see the soft shadows have penumbra regions. Figure 8(c) and Figure 20 show the soft shadows of the Bonsai tree dataset and the mushrooms. In Figure 20, we can notice a higher degree of blur in the shadows of the regions farther from the plane, like the shadows of the top part of the tree, while the shadows of the two small components in front of the tree look pretty solid, since they are close to the plane. These soft shadows look more realistic than the hard shadows in Figure 10, Figure 8(b) and figure 9.

6 Discussion

The rendering time using hardware-based 3D texture mapping depends on the size of the dataset and the number of the slices. All performance data is obtained using a PC equipped with a Pentium 4 processor of 3.2 GHz and a Quadro FX 3000 graphics card with 256 MB graphics memory. For the Bonsai tree dataset (256^3) and the polygonal mushrooms in Figure 8, which are sliced with the interval at voxel spacing (256 slices for orthogonal and 442 slices for diagonal direction) and rendered to 512x512 images, the rendering time without shadows is 410ms, and the rendering time with shadows is 570ms. The calculation of shadows takes 39% longer rendering time. For the implementation of soft shadows, the rendering time is much longer (185s), due to the expensive convolution calculation. Currently, we create soft shadow by convolving all pixels in the shadow buffer using hardware-based convolution operation. We create the shadow buffer with the same size as the image resolution (512x512). Due to the intrinsic time-consuming of convolution, soft shadow cannot be rendered in real time in our current implementation, considering the large number of convolution operations for each slice. On the other hand, the resolution of shadow buffer can be much smaller than the image size without the significant loss of shading accuracy. This would improve the performance of soft shadow generation with 3-4 times speedup if a shadow buffer with half size of image resolution is used.

Since the volumetric dataset is loaded into graphics hardware memory as a 3D texture, the size of volume is limited by the texture memory on graphics hardware. The volume size could be as large as 512x512x512 (128 MB) to be loaded on our Quadro FX 3000 graphics card.

Given a fixed light in eye space, our slicing scheme is not changed, in which all slices are either perpendicular to the halfway vector of light and eye vectors if light and eye are at the same side of the volume or parallel to the halfway vector if light and eye are the different side of the volume as in [13]. This static slicing scheme guarantees no visual artifacts like popping [34]

due to switching slicing direction occur. On the other hand, if the light position is time-varying in eye space, the slicing direction will be changed accordingly. The popping artifacts would occur if the light position changes abruptly and leads to the significant change of slicing direction for the neighboring frames.

7 Conclusions

In this paper, we first describe how to implement multiple scattering and incorporate multiple scattering with our light attenuation model. We use a convolution technique to approximate the multiple forward scattering and back scattering for clouds, a high albedo participating medium.

Based on the shadow and soft shadow algorithm for volumetric data [32,33], this paper extends the algorithm to generate shadows and soft shadows for scenes including both volumes and polygons. The polygons are first rendered with respect to both the eye and the light source, and the depth information is retrieved. Texture-based rendering is then used to render the volumes. During the volume rendering, polygons are composited into the volumes slice by slice in a depth-sorted order. We have implemented our shadow algorithm to include soft shadows combining volumes and polygons. This shadow algorithm handles all combinations of volumes and polygons, without any restriction on the geometric positioning and overlap of the volumes and polygons.

Now our shadow algorithm can generate shadows or soft shadows for point lights, parallel lights, projective textured lights and extended light sources [32,33]. Also, our algorithm can deal with both volumes (including volumetric datasets and hypertextured objects) and polygons, and combine multiple scattering and light attenuation model. Our shadow algorithm is a complete system for shadow generation and provides we believe the first such system for hardware-based volume rendering.

Acknowledgments

We would like to thank the NSF Career Award (#9876022) for support to this project, and thank S. Roettger, VIS, University of Stuttgart for providing the Bonsai tree dataset in Figure 8 and the University of Erlangen-Nuremberg for providing the Bonsai tree dataset in Figure 9 and Figure 20.

References

[1] P. Atherton, K. Weiler, D. Greenberg, "Polygon Shadow Generation", *Proc. SIGGRAPH'78*, pp. 275-281, 1978.

[2] S. Chandrasekhar, *Radiative Transfer*, Oxford University Press, 1950.

[3] F. Crow, "Shadow Algorithm for Computer Graphics", *Proc. SIGGRAPH'77*, pp. 242-248, 1977.

[4] Y. Dobashi, T. Nishita, T. Yamamoto, "Interactive Rendering of Atmospheric Scattering Effects Using Graphics Hardware", *Proc. Graphics Hardware 2002*, pp.99-108, 2002.

[5] Y. Dobashi, T. Yamamoto, T. Nishita, "Interactive Rendering Method for Displaying Shafts of Light," *Proc. Pacific Graphics 2000*, pp.31-37, 2000

[6] D. Ebert, R. Parent, "Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques", *Proc. SIGGRAPH'90*, pp. 357-366, 1990.

[7] M. Harris, A. Lastra, "Real-Time Cloud Rendering", *Proc. Eurographics'2001*, vol. 20, no. 3, pp. 76-84, 2001.

[8] H.W. Jensen, S.R. Marschner, M. Levoy, P. Hanrahan, "A Practical Model for Subsurface Light Transport", *Proc. SIGGRAPH'01*, pp. 511-518, 2001.

[9] J. Kajiya, B. Von Herzen, "Ray Tracing Volume Densities", *Proc. SIGGRAPH'84*, pp. 165-174, 1984.

[10] A. Kaufman, "An Algorithm for 3D Scan-Conversion of Polygons", *Proc. Eurographics'87*, pp. 197-208, 1987.

[11] J. Kniss, G. Kindlmann, C. Hansen, "Multi-Dimensional Transfer Function for Interactive Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 270-285, 2002.

[12] J. Kniss, S. Premoze, C. Hansen, D. Ebert, "Interactive Translucent Volume Rendering and Procedural Modeling", IEEE Visualization 2002.

[13] J. Kniss, S. Premoze, C. Hansen, P. Shirley, A. McPherson, "A Model for Volume Lighting and Modeling", *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 150-162, 2003.

[14] K.D. Lathrop, "Ray Effects in Discrete Ordinates Equations", *Nuclear Science and Engineering*, vol. 32, pp. 357-369, 1968.

[15] M. Levoy, "A Hybrid Ray Tracer for Rendering Polygon and Volume Data", *IEEE Computer Graphics and Applications*, vol. 10, no. 2, pp. 33-40, 1990.

[16] T. Lokovic, E. Veach, "Deep Shadow Map", *Proc. SIGGRAPH'2000*, 2000.

[17] W.E. Lorensen, H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Computer Graphics*, vol. 21, no. 4, pp. 163-169, 1987.

[18] N. Max, "Optical Models for Direct Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99-108, 1995.

[19] N. Max, "Efficient Light Propagation for Multiple Anisotropic Volume Scattering", *Photorealistic Rendering Techniques*, G. Sakas, P. Shirley, and S. Mueller, eds. Heidelberg: Springer Verlag, pp.87-104, 1995.

[20] R. Miyazaki, Y. Dobashi, T. Nishita, "A Fast Rendering Method of Clouds Using Shadow-View Slices" *Proc. CGIM 2004*, pp.93-98, 2004

[21] T. Nishita, E. Nakamae, "An Algorithm for Half-Tone Representation of Three-Dimensional Objects", *Information Processing in Japan*, vol. 14, pp. 93-99, 1974.

[22] T. Nishita, E. Nakamae, "Half-Tone Representation of 3-D Objects Illuminated by Area Sources or Polyhedron Sources," *IEEE Computer Society's 7th International Computer Software & Applications Conference (COMPSAC)*, pp.237-242, 1983

[23] T. Nishita, Y. Dobashi, E. Nakamae, "Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light", *Proc. SIGGRAPH'96*, pp. 313-322, 1996.

[24] M. Nulkar, K. Mueller, "Splatting With Shadows", *Volume Graphics 2001*.

[25] K. Perlin, E. M. Hoffert, "Hypertexture", *Proc. SIGGRAPH'89*, pp. 253-262, 1989.

[26] H. Rushmeier, K. Torrance, "The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium", Computer Graphics, vol. 21, no. 4, pp. 293-303, 1987.

[27] H. Rushmeier, "Realistic Image Synthesis for Scenes with Radiatively Participating Media", PhD Thesis, Cornell University, May 1988.

[28] C. Soler, F.X. Sillion, "Fast Calculation of Soft Shadow Textures Using Convolution", Proc. SIGGRAPH'98, pp. 321-332, 1998.

[29] T. Whitted, "An Improved Illumination for Shaded Display", *Communications of the ACM*, Vol. 23, No. 6, pp. 343-349, 1980.

[30] L. Williams, "Casting Curved Shadows on Curved Surfaces", *Proc. SIGGRAPH'78*, pp. 270-174, 1978.

[31] A. Woo, P. Poulin, A. Fournier, "A Survey of Shadow Algorithm", *IEEE Computer Graphics and Applications*, vol. 10, no. 6, 1990.

[32] C. Zhang, R. Crawfis, "Volumetric Shadows Using Splatting", *Proc. Visualization 2002*, pp. 85-93, 2002.

[33] C. Zhang, R. Crawfis, "Shadows and Soft Shadows with Participating Media Using Splatting", *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 139-149, 2003.

[34] Mueller, Klaus, and Roger Crawfis, Eliminating Popping in Sheet Buffer-Based Splatting, *IEEE Visualization '98,* 1998.