Light Propagation for Mixed Polygonal and Volumetric Data Using Splatting

Caixia Zhang, Roger Crawfis
Department of Computer and Information Science
The Ohio State University, Columbus, OH

Abstract

This paper describes a shadow and soft shadow algorithm for mixed polygonal and volumetric data. The polygons are first rendered using scanline rasterization and the depth information is retrieved. Volume rendering using splatting is then used to render the volumes. Polygons are composited with volumes during the volume rendering in depth-sorted order slice by slice. We implement shadow and soft shadow algorithms, combining volumes and polygons using sheet-based splatting. This shadow algorithm applies to all combinations of volumes and polygons, without any restriction on the geometric positioning and overlap of the volumes and polygons. This paper also discusses how to implement multiple scattering for high-albedo participating media during the shadow generation using sheet-based splatting. We use a convolution technique to model the multiple forward and back scattering and show results for clouds, a high-albedo participating medium. Our algorithm constitutes a complete system for shadow generation.

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

Volume rendering is the display of datasets sampled in three dimensions. Splatting is one volume rendering algorithm, which can create high-quality images, and render efficiently in the case of a sparse dataset. The basic principles of a splatting algorithm are: (1) represent the volume as an array of overlapping basis functions with amplitudes scaled by the voxel values; (2) project these basis functions to the screen to achieve an approximation of the volume integral [4]. A major advantage of splatting is that only relevant voxels are projected and rasterized. This can tremendously reduce the volume data that needs to be processed and stored.

A shadow is a region of relative darkness within an illuminated region caused by an object totally or partially occluding the light. Shadows are essential to realistic images. Earlier implementations of shadows focused on hard shadows, in which a value of 0 or 1 is multiplied with the light intensity. In volume rendering, as the light traverses the volume, the light intensity is continuously attenuated by the volumetric densities. We have proposed a shadow algorithm that properly determines this light attenuation and generate shadows from volumetric datasets using splatting [35].

Soft shadows include an umbra region, areas for which no part of the extended light source is visible, and a penumbra region, areas in which part of the extended light source is visible and part is hidden or occluded. The generation of soft shadows is a difficult topic in computer graphics. It requires integrating the contributions of extended light sources on the illumination of objects. We have proposed a soft shadow algorithm with a convolution technique using sheet-based splatting [36].

Some visualization applications require the volumetric and geometrical objects to appear together in a single image. For example, some geometrically defined objects are surrounded by clouds, smoke, fog, or other gaseous phenomena. The above shadow and soft shadow algorithm generates shadows for volumetric datasets only. In this paper, we will extend our shadow and soft shadow algorithm to deal with a scene including both volumetric datasets and polygonal geometries.

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

In the following section, background and previous work are 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.

2. PREVIOUS WORK

2.1 Splatting

In splatting, each voxel is represented by a 3D kernel weighted by the voxel value. The 3D kernels are integrated into a generic 2D footprint along the traversing ray from the eye. This footprint can be efficiently mapped onto the image plane and the final image is obtained by the collection of all projected footprints, weighted by the voxel values. This splatting approach is fast, but it suffers from color bleeding and popping artifacts due to incorrect volume integration.

Mueller et al. [23] eliminates this popping drawback by aligning the sheets to be parallel to the image plane. This splatting method is called image-aligned sheet-based splatting. All the voxel kernels that overlap a slab are clipped to the slab and summed into a sheet buffer. The sheet buffers are composited front-to-back to form the final image. While this significantly improves image quality, it requires much more compositing and several footprint sections per voxel to be scan-converted. Splatting using post classification was proposed by Mueller et al. [21] to generate images with crisp edges and well-preserved

{zhangc,crawfis}@cis.ohio-state.edu, 2015 Neil Ave., 395 Dreese Lab, Columbus, OH 43210, USA

surface details. In this paper, we use the post-classification to keep track of the per-pixel contribution to the light attenuation and generate per-pixel shadows.

2.2 Shadow Algorithms

Earlier implementations of shadows focused on hard shadows, in which a value of 0 or 1 is multiplied with the light intensity. The shadow volume algorithm by Crow [5] introduces the concept of shadow volumes. A 2-pass hidden surface algorithm is proposed by Nishita and Nakamae [24] and Atherton et al. [1]. Williams [33] uses a z-buffer depth-map algorithm to generate shadows.

These shadow algorithms can only determine if an object point 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 the volume, the light intensity is continuously scattered and attenuated by the volumetric densities. 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 [32] and volumetric datasets [6,10]. However, the volumetric shadows using ray tracing are very costly computationally [19].

For splatting, Nulkar and Mueller have implemented an algorithm to add shadows to volumetric scenes using splatting [26]. They use a two-stage splatting approach. In the first-stage, splatting is used to construct a three-dimensional light volume; the second stage is formed by the usual rendering pipeline. Since the algorithm needs a 3D buffer to store the light volume, it has the problem of high storage and memory cost. Also, accurate shadows are difficult to implement using this method, due to the limited resolution of the light volume. We investigate a new algorithm to implement shadows using splatting that requires only a 2D buffer for each light source [35,36]. Kniss et al. [12] also utilize an off screen buffer to accumulate the light attenuation in their volume rendering using 3D texture slicing. In this paper, we will extend our shadow algorithm to generate shadows for a scene including both volumetric datasets and polygons.

2.3 Multiple Scattering

For low albedo participating media, the scattering is unimportant compared to the light attenuation. So, the two-pass algorithm proposed by Kajiya and Von Herzen [10] is used to calculate the illumination. The first pass computes the light intensity reaching each voxel, and in the second pass, the light is reflected or scattered to the viewpoint. This two-pass method is a single scattering model and it is only valid for low albedo media. Multiple scattering is important for realistic rendering of high albedo participating media, for example, clouds. Multiple scattering must account for scattering in all directions. It is more physically accurate, but much more complicated and expensive to evaluate. Max [17] gives an excellent survey of optical models, including multiple scattering.

The calculation of multiple scattering can be divided into four methods [17]: the zonal method, the Monte Carlo method, the P-N method and the discrete ordinates method. In the zonal method [28], 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 [29], 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 [2,10] 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 [18] 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. They use impostors to accelerate cloud rendering. Kniss et al. [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. Our motivation is to implement multiple scattering using splatting and add multiple scattering to our shadow algorithm so that we can deal with high albedo participating media, like clouds.

2.4 Rendering both Volumes and Polygons

Since some visualization applications require volumetric and geometrical objects to appear together in a single image, a volume rendering technique which incorporates objects described by surface geometries is necessary to render both surface geometries and volume modeled objects.

The most common solution is to convert polygonal and volumetric data into a common representation: either construct surface polygons from volume data [16] or change polygon data to volume data using 3D scan-conversion [11]. 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 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 techniques [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 the gaseous phenomena. From a review of the literature, splatting has not been extended to render both volumes and polygons. The motivations of our work include extending the sheet-based splatting algorithm to incorporate polygon modeled objects and generating shadows and soft shadows for scenes with both volumes and polygons.

3. MULTIPLE SCATTERING

Our shadow algorithm models the light attenuation as light traverses the volume. It has generated realistic shadows for low albedo participating media. However, 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 splatting and incorporate it with our shadow algorithm by displaying clouds, a high albedo participating medium. The clouds are modeled as a collection of ellipsoids, and Perlin's fractal function [27] is used to disturb the

density distribution. Figure 1 shows the clouds with the light emanating behind it. The clouds look unnaturally dark, because only light attenuation is modeled.

3.1 Implementation of Multiple Scattering

Considering multiple scattering, the light intensity at a point ***P*** is the sum of the direct light from the light source that is not absorbed by intervening particles and the light scattered to ***P*** from other particles. The calculation of multiple scattering requires accounting for scattering in all directions. Therefore, $I(P,\omega)$, the intensity at each point P in each light flow direction ω, 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), and 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, 17], 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.

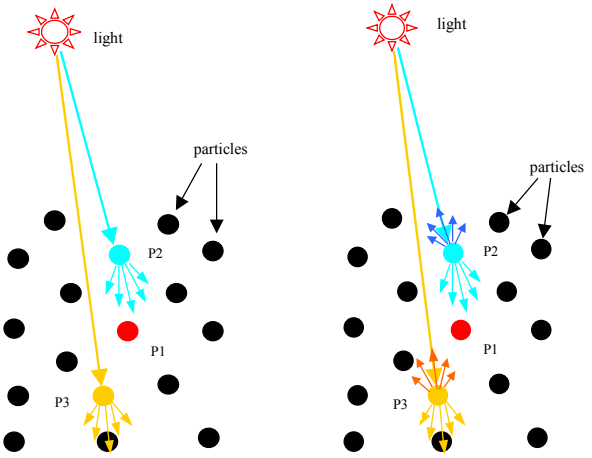
A full multiple scattering algorithm must compute this quantity for all light flow directions. This would be very expensive and impractical with sheet-based splatting. Nishita et al. [25] take advantage of the strong forward scattering characteristics of clouds and limit the sampled light flow directions to sub-spaces of high contribution. Harris et al. [7] further approximate multiple forward scattering only in the light direction and get good cloud images. In this paper, we model both multiple forward scattering and back scattering and we approximate the multiple scattering along the light direction based on the strong forward scattering characteristics of clouds. Now we can 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 ***l***. In sheet-based splatting, we can calculate the integration sheet by sheet. We then get the following formulas:

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

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

The above g_k^f represents the forward scattering and g_k^b is for the back scattering. γ_A and γ_B are the 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. ***k*** means the values are from the sheet ***k***.

Since the distribution of $I(x,l)$ is encoded within a sheet, 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 kernel can be selected to account for the coefficient $a_k(x)\cdot\tau_k(x)\cdot p(l,l_{\gamma_A})$ or $a_k(x)\cdot\tau_k(x)\cdot p(l,l_{\gamma_B})$. The convolution calculation is easy to implement in sheet-based splatting. This shows that multiple scattering can be modeled as light diffusion along the light direction. Kniss et al. [13] also use convolution technique to model the light diffusion, but they just use an empirical volume shading model without theoretical derivation and they only model multiple forward scattering.



(a) (b)

Figure 3: A schematic of light transport (a: forward scattering; b: forward scattering and back scattering)

If only multiple forward scattering is considered, we can get 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 sheet ***k-1***. The recurrence relation says the light incident on sheet ***k*** is equal to the intensity scattered to sheet ***k*** from the sheet ***k-1*** plus the direct light transmitted to the sheet ***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 sheet, then calculate the multiple forward scattering g_{k-1} using convolution over I_{k-1} and add it to I_k. Figure 2 shows the clouds with multiple forward scattering only. The clouds look brighter than the clouds in Figure 1 without multiple scattering.

From the physical viewpoint, when the light is scattered forward, some light will scatter backwards. In Figure 3a, P1 receives energy from P2 by forward scattering, while in Figure 3b, P1 also gets energy from P3 by back scattering. In order to implement back scattering, we keep the light intensity of the sheets which contribute to the current sheet ***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 sheet obtains new energy from back scattering, it will add this to the stored light intensity, and then bounce back energy to its upper sheet. This process continues until the sheet ***k*** to be illuminated is reached. After the illumination, the total energy at the sheet ***k*** is scattered forward to sheet ***k+1***.

If we add back scattering to the clouds in Figure 2, we get the clouds in Figure 4. As energy bounces around, the edges are brighter, as is the interior region of the clouds.

Figure 1: Clouds without multiple scattering

Figure 2: Clouds with multiple forward scattering only

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

4. SHADOW ALGORITHM COMBINING VOLUMES AND POLYGONS

Our previous work developed a shadow algorithm for volumetric data using sheet-based splatting [35]. In this paper, we will extend the shadow algorithm to generate shadows for both volume and polygon data. We assume the polygons are opaque in this section to describe the shadow algorithm.

4.1 Shadow Algorithm

Our shadow algorithm for mixed polygonal and volumetric data has two stages. First, the polygons are rendered with respect to both the viewer and the light source, and the polygons are stored. At the second stage, sheet-based splatting is used to render the volumes, and any relevant polygon information is composited into the scene slice by slice. 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.

We use a multi-pass algorithm. Our first pass is to render the polygons. Since our purpose is to generate shadows, the polygons are scanline rasterized with respect to 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 final rendering. For the image generation, we store the object color, the normal and the z value with respect to the viewer into three buffers aligned with the viewer. For the shadow generation, we save the z value with respect to the light into a z-buffer aligned with the light source.

The second stage is to render the volumes and composite the polygon information slice by slice. Similar to the shadow algorithm for volumes only [35], we use the non-image-aligned sheet-based splatting (as shown in Figure 5). The volume is sliced along the half-way vector between the eye vector and the light vector. The image buffer is aligned with the eye, and the shadow buffer is aligned with the light source.

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.

The polygons at different regions are processed in different ways. The polygons in region 1 are rendered in one step using the information stored in the buffers, and whether a point is in shadow or not is determined by the z value stored in the z-buffer with respect to the light. Actually, the z-buffer shadow algorithm is used for region 1. After rendering the polygons in this region, the shadow of the polygons in front of the volume is calculated for the first volumetric slice.

For region 2, whether there are polygons in the current slab is determined by the depth information of the polygon z-buffer with respect to the viewer. If there are polygons at the current slab (as shown in Figure 7), for those pixels corresponding to the polygons, the footprint evaluation will be calculated in the shaded region in Figure 7. This is defined by the starting edge of the slab and the polygon boundary. This footprint contribution is then composited to the frame buffer with respect to the eye. Next, the contribution of the polygons in this slab is composited into the frame buffer with the opacity set to 1.0. Similarly, the contribution of the footprints and the polygons on the light attenuation are composited to the shadow buffer with respect to the light source. In the region 2, the shadow value of a pixel on the sheet image buffer is determined by a corresponding pixel on the accumulated shadow buffer, as discussed in our shadow algorithm [35].

The polygons in region 3 are similar to the polygons in region 1, but 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.

The above shadow algorithm for both the volumes and polygons using sheet-based splatting 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 either region 1 or region 3.

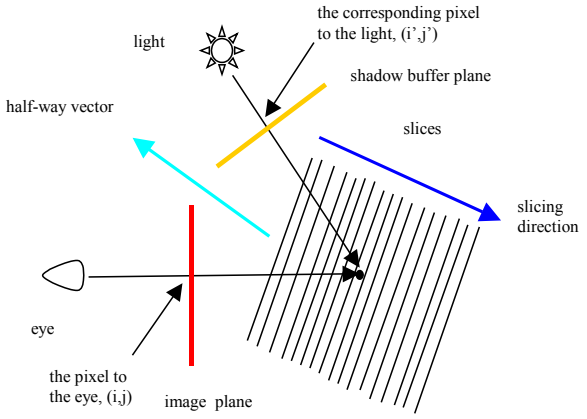


Figure 5: Non-image-aligned sheet-based splatting

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

Table 1: Shadow determination for three regions

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

The shadow algorithm combining volumes and polygons using splatting is demonstrated with the following pseudocode.

1. Rasterize the polygons wrt the eye and store the color, normal and depth;
2. Rasterize the polygons wrt the light source and store the depth;
3. Transform each voxel to the coordinate system having the half way vector as the z-axis;
4. Bucket sort voxels according to the transformed z-values;
5. Initialize opacity map to zero;
6. Initialize the shadow buffer to zero;
7. Render the polygons in region 1;
8. Add the shadows caused by polygons in region 1 to the shadow buffer;
9. For each sheet in front-to-back order
10. Initialize image sheet buffer;
11. Initialize shadow sheet buffer;
12. For each footprint
13. Rasterize and add the footprint to the current image sheet buffer;
14. Rasterize and add the footprint to the current shadow sheet buffer;
15. End for;
16. Calculate the gradient for each pixel using central differences;
17. Classify each pixel in the current image sheet buffer;
18. Map each pixel to the shadow buffer and get its opacity;
19. Calculate the illumination to obtain the final color;
20. Composite the current image sheet buffer to the frame buffer;
21. Classify each pixel on current shadow sheet buffer and composite it to the accumulated shadow buffer;
22. Calculate the illumination for polygon pixels in current sheet and composite it to the frame buffer;
23. Composite the polygon contribution in the current sheet to the accumulated shadow buffer;
24. End for;
25. Render the rest part of polygons in region 3;

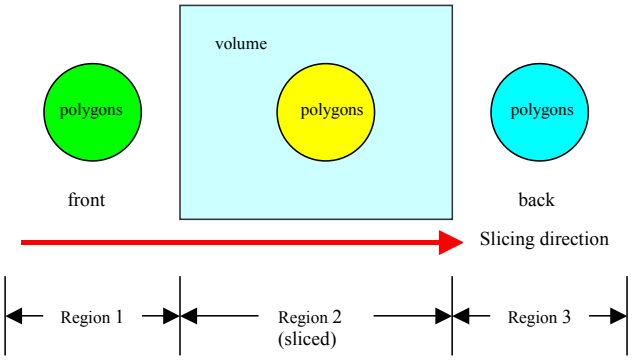


Figure 6: Position relationship between polygons and volume with respect to the slicing direction

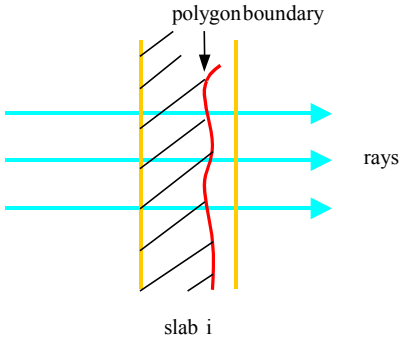


Figure 7: Region of footprint evaluation

4.2 Shadow Results

Using the above shadow algorithm, we have implemented shadows for scenes including both volumes and polygons.

Figure 8 shows the shadows cast from three mushrooms. The mushrooms are polygons and the bottom plate is represented as volumetric data. Figure 9 shows a scene in which a dart is thrown at some rings. The dart is a polygon model, and the rings and the plate are volumetric data.

In Figure 10, some mushrooms are under the volumetric Bonsai tree. The mushrooms cast shadows on the bottom plate and three mushrooms are in the shadows of the Bonsai tree.

Figure 11 and Figure 12 are two scenes in which polygons are inside semi-transparent volumetric data. Figure 11 shows a scene of a polygon teapot inside volumetric translucent media. We can see the shadows cast by the teapot on itself and through the translucent media. Figure 12 is an example of back-to-front rendering: light comes into the room from the back. A polygon-defined desk resides in the smoky room. The smoke is modeled using Perlin's turbulence function [27].

Our shadow algorithm combining volumes and polygons also works for high-albedo media. In Figure 13, 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. We propose a soft shadow algorithm for volumetric data

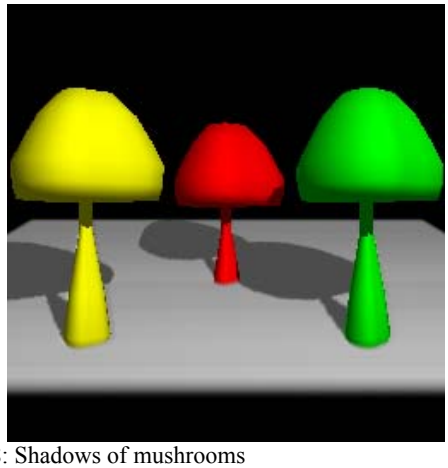


Figure 8: Shadows of mushrooms

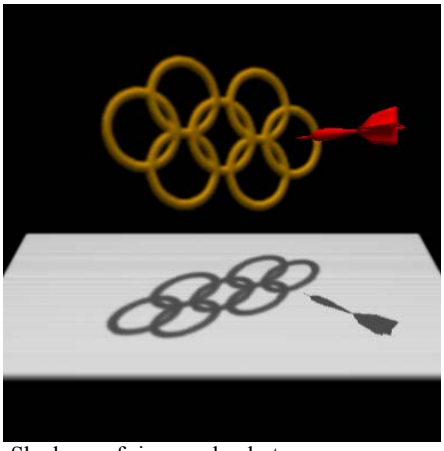


Figure 9: Shadows of rings and a dart

Figure 10: Bonsai tree and mushrooms

using sheet-based splatting in [36]. Our soft shadow algorithm is an analytic algorithm using convolution techniques. If the light source is parallel to the slices, then we can calculate the width of the penumbra region, and generate the soft shadows by convolving the shadows with respect to the center of the extended light source, which is generated using the shadow algorithm discussed in Section 4. In the soft shadow algorithm using sheet-based splatting, the volume is still sliced along the half-way vector between the eye vector and the light vector (as shown in

Figure 11: A scene of a teapot inside a translucent cube

Figure 12: A scene of a desk inside a smoky room

Figure 13: An airplane flying above the clouds

Figure 14). The image plane is aligned with the eye. Due to the requirement of our soft shadow algorithm, a virtual light source parallel to the slices is constructed and the shadow buffer is parallel to the slices (as shown in Figure 14).

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 15). So, we treat both the volumes and the polygons as a whole volume, although the polygons are rendered first and their information is stored.

Similar to the shadow algorithm in Section 4, there are two stages for the soft shadow algorithm. The first stage is to render the polygons with respect to the viewer and the light source respectively, and store the information of the polygons. At the second stage, splatting is used to render the volumes, composite the polygon information and generate soft shadows slice by slice. The first stage is exactly same as the shadow algorithm in Section

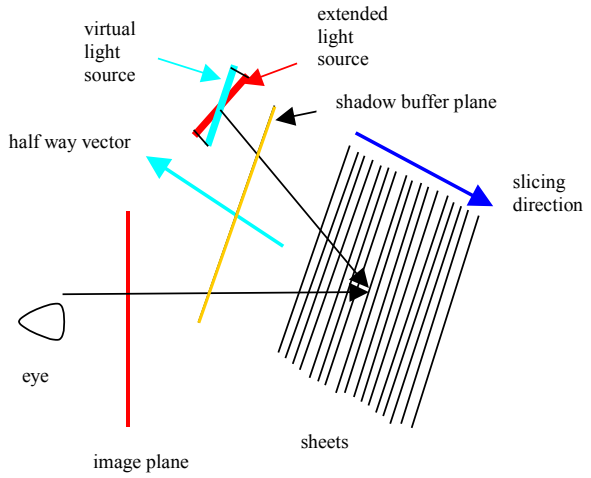


Figure 14: Non-image-aligned sheet-based splatting for soft shadows

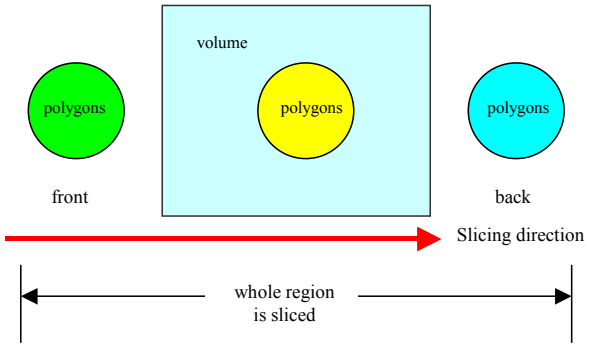


Figure 15: The whole region rendered slice by slice

4. However, there are some differences in the second stage between the shadow algorithm in Section 4 and the soft shadow algorithm in this section.

The shadow buffer is now parallel to the slices as required by our soft shadow algorithm. So, the z-buffer for polygons obtained in stage 1 should be mapped to a new z-buffer parallel to the slices for easy use in stage 2.

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, 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 part of the polygons belonging to the current slab is used to update the shadow buffer. Then the shadow buffer is convolved to prepare for the next slice. Since the shadow information of the polygons is stored in a z-buffer, we can imagine the contribution of the polygons is the boundary of solid objects. The penumbra-looking region is not the real penumbra region. The reason is that the contribution of the polygons on the shadow buffer is the boundary of the regions, not the solid region. The solution is to fill the regions defined by the boundary. Since we only keep the z-buffer, our approximate method is to estimate the thickness for the z-buffer. We use a thickness that is proportional to the reciprocal of the gradient of the z depth. In this way, we get realistic, solid-looking soft shadows, where the silhouettes imply a carved surface with respect to the light.

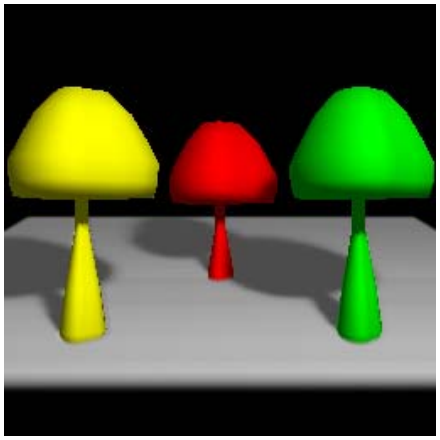


Figure 16: Soft shadows of mushrooms

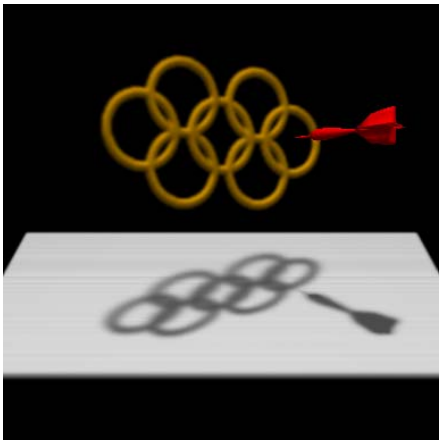


Figure 17: Soft shadows of rings and a dart

Figure 18: Soft shadows of Bonsai tree and mushrooms

We have generated some soft shadows for scenes with both volumes and polygons, using the above soft shadow algorithm. Figure 16 is the soft shadows of the mushrooms. Compared with the hard shadows in Figure 8, the soft shadows have penumbra regions. Figure 17 shows the soft shadows of the volumetric rings and the polygon-modeled dart. Figure 18 shows the soft shadows of the Bonsai tree dataset and the mushrooms. These soft shadows

look more realistic than the hard shadows in Figure 9 and Figure 10.

6. CONCLUSIONS

In this paper, we first describe how to implement multiple scattering using sheet-based splatting 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 our shadow and soft shadow algorithm for volumetric data [35,36], this paper extends the algorithm to generate shadows and soft shadows for scenes including both volumes and polygons. The polygons are first rendered using scanline rasterization with respect to both the eye and the light source, and the depth information is retrieved. Splatting is then used to render the volumes. During the volume rendering, polygons are composited into the volumes slice by slice in the depth-sorted order. We have implemented shadow algorithm and soft shadow algorithm combining volumes and polygons. This shadow algorithm applies to all combinations of volumes and polygons, without any restriction on the geometric positioning and overlap of the volumes and polygons.

In this paper, we explain the shadow algorithm and soft shadow algorithm with the assumption of opaque polygons. Our future work is to extend the algorithm to deal with translucent polygons.

Now our shadow algorithm can generate shadows or soft shadows for point lights, parallel lights, projective textured lights and extended light sources [35,36]. 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.

7. ACKNOWLEDGMENTS

We would like to thank the NSF Career Award (#9876022) for support to this project and thank the University of Erlangen-Nuremberg for providing the Bonsai tree datasets.

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] R. Crawfis, J. Huang, "High Quality Splatting and Volume Synthesis", *Data Visualization: the state of the art*, F.H. Post, G.M. Nielson, G.P. Bonneau, eds. Kluwer Academic Publishers, pp. 127-140, 2003.

[4] R. Crawfis, N. Max, "Texture Splats for 3D Scalar and Vector Field Visualization", Proc. Visualization'93 , pp. 261-266, 1993.

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

[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] J. Huang, K. Mueller, N. Shareef, R. Crawfis, "FastSplats: Optimized Splatting on Rectilinear Grids", *Visualization'2000*, pp. 219-227, 2000.

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

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

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

[12] 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.

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

[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] 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.

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

[18] 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.

[19] M. Meissner, J. Huang, D. Bartz, K. Mueller, R. Crawfis, "A Practical evaluation of Popular Volume Rendering Algorithms", *2000 Symposium on Volume Rendering*, pp. 81-90, Salt Lake City, October 2000.

[20] K. Mueller, T. Moeller, J.E. Swan, R. Crawfis, N. Shareef, R. Yagel, "Splatting Errors and Antialiasing", *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 2, pp. 178-191, 1998.

[21] K. Mueller, T. Moeller, R. Crawfis, "Splatting Without the Blur", *Proc. Visualization'99*, pp. 363-371, 1999.

[22] K. Mueller, N. Shareef, J. Huang, R. Crawfis, "High-quality Splatting on Rectilinear Grids with Efficient Culling of Occluded Voxels", *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 2, pp. 116-134, 1999.

[23] K. Mueller, R. Crawfis, "Eliminating Popping Artifacts in Sheet Buffer-based Splatting", *Proc. Visualization'98*, pp.239-245, 1998.

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

[25] 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.

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

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

[28] 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.

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

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

[31] L. Westover, "Interactive Volume Rendering", *Proceedings of Volume Visualization Workshop*, University of North Carolina, Chapel Hill, N.C., 1989, pp. 9-16.

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

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

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

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

[36] 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.