Article

Approximate Image-Space Multidirectional Occlusion Shading Model for Direct Volume Rendering

Yun Jang * and Seokyeon Kim

Department of Computer Engineering and Convergence Engineering for Intelligent Drone, Sejong University, Seoul 05006, Korea; ksy0586@gmail.com
* Correspondence: jangy@sejong.edu

Abstract: Understanding and perceiving three-dimensional scientific visualizations, such as volume rendering, benefit from visual cues produced by the shading models. The conventional approaches are local shading models since they are computationally inexpensive and straightforward to implement. However, the local shading models do not always provide proper visual cues since non-local information is not sufficiently taken into account for the shading. Global illumination models achieve better visual cues, but they are often computationally expensive. It has been shown that alternative illumination models, such as ambient occlusion, multidirectional shading, and shadows, provide decent perceptual cues. Although these models improve upon local shading models, they still require expensive preprocessing, extra GPU memory, and a high computational cost, which cause a lack of interactivity during the transfer function manipulations and light position changes. In this paper, we proposed an approximate image-space multidirectional occlusion shading model for the volume rendering. Our model was computationally less expensive compared to the global illumination models and did not require preprocessing. Moreover, interactive transfer function manipulations and light position changes were achievable. Our model simulated a wide range of shading behaviors, such as ambient occlusion and soft and hard shadows, and can be effortlessly applied to existing rendering systems such as direct volume rendering. We showed that the suggested model enhanced the visual cues with modest computational costs.

Keywords: occlusion model; image-space; direct volume rendering

1. Introduction

Direct volume rendering is a conventional technique to visualize volumetric datasets, such as measured medical data and scientific simulation data. The direct volume rendering is beneficial when the rendering provides proper visual cues to emphasize the spatial relationships within the data. The visual cues can be further enhanced by improving the shading models in the rendering pipeline. The realism supported by the improved shading models enables the user to understand the data better since the rendering images become realistic, similar to a natural image. Therefore, many approaches have been proposed to enhance the conventional shading models.

On the one hand, many simplified shading models from the global illumination models [1] have been developed for interactive rendering performance, such as ambient occlusion and shadow maps, which are volumetric approaches. These techniques interactively provide perceptually better visual cues in the rendering. However, most of the techniques require long computing times for preprocessing, a substantial amount of extra GPU memory, multiple additional rendering passes, or all of these. Moreover, the rendering performance becomes significantly degraded if the additional cost scales relative to the size of the input volume data. Some of the approximations [2,3] interactively operate on-the-fly and do not require additional GPU memory. However, they are limited to a particular rendering technique such as slice-based direct volume rendering, and the additional computation is proportional to the size of the volume data.
On the other hand, to overcome the performance degradation proportional to the data size, image-space occlusion techniques have been proposed for the polygonal rendering [4–9] that is widely used in computer graphics and the gaming industry. The color, normal, and depth of a scene are rendered to buffers, and the occlusion is computed with the depth and normal buffers. The benefits of these methods are that they are independent of the rendering engine and can be applied to any rendering pipeline by merely adding a screen-space pass. Furthermore, the computation is much less compared to the volumetric approaches.

In this work, we present an approximate image-space multidirectional occlusion shading (ISMOS) model for the direct volume rendering. Our model, ISMOS, simulates the global illumination effects in a simplified manner. ISMOS utilizes the rendered depth and normal buffers for the occlusion shading computation at the end of the direct volume-rendering pipeline. ISMOS does not require preprocessing, nor extra GPU memory for the entire volume data. Since the model is operated as an additional computation in the shader, ISMOS can be easily employed in any volume-rendering pipeline. Besides, the transfer function manipulations and light property changes, including light positions, can be interactively executed on-the-fly without interrupting the rendering since ISMOS utilizes the rendered buffer information on the GPU memory. ISMOS simulates various light sources, whereas previous models simulated only ambient and directional light sources. For the sampling strategy, while previous models sampled test points within a hemisphere centered at each pixel, ISMOS takes sample points along a ray between a pixel and the light source to calculate the amount of light transferred to the pixel. Note that preprocessing indicates the light volume generation before the ray casting volume rendering, and the light volume is repeatedly generated whenever the light parameters and transfer function parameters are changed in other approaches.

Figure 1a illustrates the ISMOS model setup, including the volume-rendered image-space, eye position, and light source. The volume-rendered image-space contains the color, depth, and normal information stored in the renderbuffers. The eye position indicates the camera location, and the light source is placed on a hemisphere with the radius $R$, centered at the center of the image-space. Figure 1b,c shows three different light sources, including point, area, and ambient lights. Note that the light source in (a) is an example of the area light source with the radius $d$. Since our image-space model does not require notable extra computation time, ISMOS easily enabled us to manipulate multiple light sources. The multiple directional light sources were difficult to apply in the previous image-space approaches because the hemispheres centered at the pixels were not directional. Besides, we applied an opacity threshold for the shading effect estimation on the opaque pixels to make the inner volume structures stand out, which allowed us to create renderings accommodating multiple transparent layers with different opacity values. We also handled transparent objects efficiently by applying a transmittance approximation with a smooth function, which can be easily performed in a volume-rendering pass, as presented in Figure 2. The approximated functional form of the transmittance describes the opacity values, which allowed us to represent all opacity values over the depth with only one parameter per pixel. Figure 3 presents the rendering pipeline. We extended the conventional volume rendering with frame buffers to simulate various occlusion shading models.

Our major contributions in this work include:

- An interactive approximate image-space multidirectional occlusion shading model applicable to the conventional direct volume-rendering pipeline without preprocessing and additional GPU memory;
- The simulation of light sources, including ambient occlusion and soft and hard shadows as a unified model;
- The handling of multiple directional lights with only a minor additional overhead;
- A multi-layered view for transparent scenes.
Figure 1. Overview of the ISMOS model. (a) presents an illustration of the ISMOS model. (b,c) show different light sources that our model can simulate. The gray lines indicate the occluded paths from a pixel to the light samples, and the dark red lines are the paths without any occluded object. The light red lines are the paths passing through transparent objects.

Figure 2. The transmittance approximation for the transparent objects. (a) During the rendering, we stored an approximated functional representation of the transmittance characteristics per pixel. The transmittance is continuously decreasing along with the entire depth. Note that the transmittance approximation is executed in the first rendering pass on-the-fly. (b) The resulting image-space representation contains information on the distribution of the transmittance and the depth.

Figure 3. The rendering pipeline with the approximate ISMOS model. The conventional volume rendering is extended with frame buffers to simulate various shading models.

2. Related Work

Volume rendering has been widely used to explore volumetric datasets and provides useful insights into the volumetric datasets. For a better understanding of the data, illumination models have been intensively studied over the decades. Various optical approximation models, such as emission–absorption models \[10\], have been widely employed to build up the volume rendering system. Levoy \[11\] presented a local illumination model with gradients in the volume rendering, which enhanced the visual cues.
However, the locally illuminated volume rendering might not give sufficient depth cues that enable unobstructed visualization of complicated features in the volumetric data. Kniss et al. [12] proposed a simple and effective interactive shading model that captured volumetric light attenuation effects, including volumetric shadows. They utilized a direct and indirect light contribution to render a translucent material. Hernell et al. [13] achieved an interactive global light propagation in the direct volume rendering. Their technique was nearly physically correct, unlike the model by Kniss et al. [12]. Tarini et al. [14] presented ambient occlusion and edge cueing for molecular visualizations. Their approach interactively enhanced the visual cues for the understanding of complex molecular surface renderings.

Ambient occlusion has been applied in volume rendering to improve visual quality and understanding. Díaz et al. [15] used summed-area tables to compute the ambient occlusion and halos. They presented a vicinity occlusion map (VOM) and a density summed-area table (DSAT) to enhance depth perception for the volumetric data. However, the VOM works in limited circumstances, and the DSAT for the volume data requires extra 3D texture memory with a long preprocessing time. Local ambient occlusion in the volume rendering was proposed by Hernell et al. [16]. The approach is a gradient-free shading model and, therefore, less sensitive to noise. In their work, an extra 3D texture was generated to hold the emissive color and the ambient light of each voxel. However, the extra 3D texture was regenerated whenever the transfer function was changed. Recently, advanced illumination models [17–24] for the volume rendering have been introduced. Ropinski et al. [23] presented an interactive volume rendering with ambient occlusion and color bleeding. Lindemann and Ropinski [19] showed advanced light material interactions. Moving light sources [21,22] have also been applied in the interactive volume rendering. In the work of [21,22], the volumetric lighting was simulated for the scattering and shadowing. However, all of these studies utilized extra 3D texture memory to store the new volumetric lighting information, which required additional preprocessing operations. Since the rendering only needed to fetch the 3D lighting textures during the rendering, the rendering speed was comparatively faster than models with on-the-fly lighting computation.

Papaioannou et al. [25] proposed a volume-based ambient occlusion model. They presented a view-independent approach by calculating ambient occlusion using normal and depth buffers with multiple volume textures. Tikhonova et al. [26] showed an enhanced visualization using a proxy image where a depth proxy was used to increase the visual cues. A directional occlusion shading model was proposed by Schott et al. [2], who presented a model for ambient occlusion within the slice-based direct volume rendering. The model was designed with conical subsets by positioning a light in front of the camera. As an extension to this work, Šoltészová et al. [3] presented a multidirectional occlusion with elliptical kernels. However, these two models worked only with the slice-based direct volume rendering and could not be applied to other volume rendering systems, such as ray casting.

Global illumination models, such as shadows [27], have been studied in many areas, including surface rendering and direct volume rendering. Hadwiger et al. [27] presented an efficient shadow rendering model with precomputed shadow maps for the direct volume rendering. Their approach required extra shadow textures to generate the final shadowed volume rendering. Annen et al. [28] employed an exponential shadow map to achieve fast shadow renderings with fewer artifacts. Banks and Beason [29] proposed a 4D light transport model to decouple the illumination computation from the isosurface generation. Recently, ambient volume scattering [30] has been proposed to simulate scattering on mesosopic scales in direct volume rendering, and an advanced lighting design [31] for the global illumination in volume rendering was presented. However, these approaches do not interactively reflect the changes of volume rendering settings, such as the transfer function and lighting properties. Zhang and Ma [32] extended their previous paper [31] to compute the light volume more interactively with CUDA and achieved faster global
illumination. However, the light volume size affected the computation speed exponentially, and the technique did not support interactive hard shadows.

Another important class of illumination techniques includes screen-space shading models [4–9,33] that have been widely used in the polygonal surface rendering and gaming industry. However, they have not been applied to volume rendering. Luft et al. [7] presented unsharp masking of the depth buffer to enhance the perceptual quality of images. Simple approximations of the ambient occlusion in the screen-space were proposed in various polygonal surface renderings [4–6]. These approaches compute the occlusion based on the depth and normal by sampling test points inside a hemisphere centered at each pixel. The rendering performance was excellent, but there were some limitations, such as the poor quality due to the undersampling, data quantization, and missing values. In order to remedy the problems, they proposed randomized sampling and blurring techniques. Nichols et al. [8] introduced an interactive multiresolution image-space rendering model for the dynamic area lighting. Timonen and Westerholm [33] applied dynamic height fields for the self-shadowing. Ritschel et al. [9] presented a screen-space directional occlusion model generalized from the screen-space ambient occlusion. They accounted for the direction of the incoming light and included one bounce of indirect illumination with minor extra computing time. The occlusion was computed with the visible directions only, in contrast to the screen-space ambient occlusion, but it was still based on a hemisphere at each pixel.

3. Image Space Multidirectional Occlusion Shading Model

In this section, we introduce our ISMOS model. The ISMOS model was designed to provide rich visual cues with a small amount of computing and memory overhead so that it could be scaled without loss of interactions and a lack of GPU memory. We present the details of our approach in the rest of this section.

3.1. Algorithm Overview

As mentioned earlier, the ISMOS model is defined in the image-space, and there is no need to change the rendering pipeline to apply the model in the existing system since the model is decoupled from the rendering system. The ISMOS model utilizes only color and depth buffers to compute the amount of light transported to a pixel. Therefore, a scene in the rendering system is simply rendered to the color and depth buffers.

When the color and depth buffers are obtained, a light source is placed on a hemisphere centered at the center of the buffers, as illustrated in Figure 1a. Each light source is designed as a circular shape with the normal vector facing towards the center of the buffers. By varying the size and position of the light source, different light types such as point, area, or ambient light can be simulated. The light source is discretized as a finite number of light samples. We computed the amount of occluded light along paths from the light samples to each pixel on the rendered buffer.

In this work, we computed the approximate amount of occluded light for opaque and transparent materials. Full occlusion was associated with the opaque material and calculated by counting the number of light samples occluded on paths from the light samples to each pixel. In contrast, partial occlusion was associated with the transparent material and computed by estimating the intensity of the light arriving at a pixel from the light samples. The details of these computations are presented in Sections 3.3.1 and 3.3.2 for the full and partial occlusion, respectively. The depth buffer was utilized to determine whether the paths were occluded by materials in the volume, which is illustrated in Figure 1b,c. In the figure, the gray paths are fully occluded by object materials in the scene. The faded pink paths are partially occluded and pass through the transparent material, which is the green object in (b). Finally, the rendering is shaded by combining the color buffer with the amount of light transmitted to the pixels in the color buffer. Note that the amount of transmitted light was computed as $1.0 - \text{occlusion}$. 
3.2. Light Model

Figure 1a illustrates the ISMOS model. The color and depth buffers were rendered as rectangles whose texture coordinates were between (0, 0) and (1, 1). The z values from the depth buffer ranged between zero and one. Note that we used 1.0—z—as the depth value in this work. All units in this section were relative to the texture coordinates. We then placed a hemisphere with a radius \( R \) at the center of the rectangle, which was \((0.5, 0.5, 0.0)\). The light source was defined as a disk whose radius was \( d \), and its center was located on the hemisphere. We then sampled \( N \) light points uniformly on the disk. \( L_{sp} \) indicates the light sample point in the light source.

The light sample points were obtained by rotating the sample points on the disk with the radius \( r \), whose center was at \((0.5 - R, 0.5, 0.0)\), to simplify the calculations. We rotated the light sample points by angle \( \phi \) about the y-axis \((R_y(\phi))\) and then rotated them once more by angle \( \theta \) about the z-axis \((R_z(\theta))\). The rotation matrix was obtained as follows.

\[
R_z(\theta)R_y(\phi) = \begin{pmatrix}
\cos(\theta)\cos(\phi) & -\sin(\theta) & -\cos(\theta)\sin(\phi) \\
\sin(\theta)\cos(\phi) & \cos(\theta) & -\sin(\theta)\sin(\phi) \\
\sin(\phi) & 0 & \cos(\phi)
\end{pmatrix}
\]

where \( 0 \leq \theta \leq 2\pi \) and \( 0 \leq \phi \leq \pi \). Note that whenever the light source moved, we just needed to compute the light sample point positions only once for all pixels. However, all previous image-space approaches generate similar samples per pixel since a hemisphere is defined per pixel. In our model, it was possible to use a sphere instead of the hemisphere for the light source location in Figure 1a. Then, the center of the sphere was \((0.5, 0.5, 0.5)\), and \( \phi \) was between zero and \( 2\pi \).

We also defined the light intensity, \( I_L \), assuming that the intensity at all light sample points from one light source was the same, to simplify computations. The intensity was attenuated by the distance between the pixel and the center of the light source. It was proportional to the inverse of the distance square as follows. \( I_L \propto \frac{1}{(L_c - P)^2} \), where \( L_c \) is the center of the light source and \( P \) is the pixel position. Note that \( P \) is represented as \( P = (P_x, P_y, z(P_x, P_y)) \), where \( z(P_x, P_y) \) is the depth value at a position \((P_x, P_y)\) on the rendered image. We could manage the light intensity of each light source.

3.3. Occlusion Computation

The occluded path between a pixel, \( P \), and a light sample point, \( L_{sp} \), implies that the material between them obstructs the light transmittance either fully or partially. Full occlusion occurs when the material completely blocks the light transmittance on the paths between the pixel and the light sample points, which means that the material is fully opaque. On the other hand, partial occlusion appears when the material is transparent. The amount of the transmitted light intensity to the pixel diminishes as the light sample travels through the transparent material along the path. Generally, the full occlusion computation is faster than the partial occlusion computation.

3.3.1. Full Occlusion

In order to compute full occlusion, we generated paths from a pixel \((P)\) to the light samples \((L_p)\) and tested whether the paths were occluded by neighbors during ray marching. The ray marching is illustrated in Figure 4a and performed on the paths defined by:

\[
r(t) = P \cdot (1 - t) + L_{sp} \cdot t,
\]

where \( t \) is between zero and one. In order to test whether \( r(t) \) passes through any opaque rendered scene, we sampled \( N_R \) test points (green dots in Figure 4a) along the path \( r(t) \). We incremented \( t \) by \( \Delta t = 1.0 / N_R \) from \( P \) and checked if the test point was occluded. For more efficient sampling of the test points, we calculated \( t_b \) at the boundary of the space and used it instead of 1.0. Therefore, the increment was rewritten as \( \Delta t = t_b / N_R \), where \( t_b < 1.0 \). The occlusion was tested by comparing the z-value of \( r(t) \) with the z-value from
the depth buffer at a pixel onto which \( r(t) \) were projected. If the z-value from the depth buffer was higher than the z-value of \( r(t) \), the path \( r(t) \) was occluded; otherwise, the path was not blocked by any rendered scene. If the test point was occluded, we then terminated the ray marching and increased \( N_o \) by one. Otherwise, we moved to the next test point by adding \( \Delta t \) to \( t \). Finally, the occlusion from the light source was computed as:

\[
O_{cc} = \frac{N_o}{N},
\]

where the \( N_o \) is the number of occluded light samples and \( N \) is the total number of light samples introduced in Section 3.2.

Once the \( O_{cc} \) was calculated, the final color was computed as:

\[
C_{\text{final}} = C_p \cdot I_L \cdot \max((1.0 - O_{cc}), I_{\text{amb}})
\]

where \( C_p \) is a color at a pixel from the color buffer assuming that the color of the light source is white and \( I_L \) is the light intensity. \( I_{\text{amb}} \) is an ambient light term that adjusts the darkness of the occlusion and prevents completely dark pixels. An example of full occlusion is presented in Figure 5b. The effect of large \( I_{\text{amb}} \) is shown in (c).

Figure 4. Comparison of uniform (a) and nonuniform (b) sampling along the path \( r(t) \). The gray lines indicate the occluded paths. Coarse sampling might not be able to find the occlusion as shown in (a), where the third path is supposed to be occluded. The green dots are the occlusion test points, and the blue dots are the light samples.

Figure 5. Different occlusion effects on the test data. Local illumination is applied to the opaque object in (a). Full occlusion (FO) is used in (b–c), and partial occlusion (PO) is shown in (d–f). \( \alpha_{th} \) is used in (f) to apply ISMOS only on an opaque blue object. \( \alpha_{app} \) indicates the approximated opacity. \( \alpha_{th} \) is a threshold alpha value, which is used in Section 3.6.

### 3.3.2. Partial Occlusion

For partial occlusion, we could use an opacity buffer in addition to the depth buffer in order to compute how much the light intensity is transported through a transparent medium, similarly to the front-to-back ray-casting computation starting the ray marching from a pixel \( P \). This additional opacity buffer is not necessary if we use the opacity approximation, which is introduced in Section 3.5, since there is only one parameter required per pixel and it can be stored in the A component of the color buffer (RGBA).
Once the ray hit a transparent medium, we accumulated the amount of occlusion for the \( i \)th light sample as follows.

\[
O_{cci} = (1 - O_{cci}) \cdot \alpha_t + O_{cci},
\]

where \( \alpha_t \) is the opacity value at \( r(t) \) and \( O_{cci} \) is initialized as zero. We stored multiple depth values per pixel in the depth buffers to handle transparent layers. In this paper, we used four front-most depth layers since our main interest lied between the front transparent layer and the back opaque layer. More detail about the multiple depth layers is described in Section 3.6. The termination conditions for the ray marching were: (i) \( r(t) \) in Equation (1) was not placed between \((0,0,0)\) and \((1,1,1)\); (ii) \( O_{cci} \) was one; and (iii) \( t \) was one. The total occlusion from \( N \) light samples was computed as follows.

\[
O_{cc} = \sum_i O_{cci}.
\]  

The final color was computed using Equation (3). Figure 5d presents an example of partial occlusion. The occlusion was computed on the front layer in the example; therefore, the occlusion was not visible behind the transparent object.

3.4. Shadow and Ambient Occlusion

In Section 3.2, we present our light source model, which was defined as a disk with a radius \( d \). The light was placed on a hemisphere with a radius \( R \), as illustrated in Figure 1a. This model allowed us to simulate different shading effects, such as shadow and ambient occlusion. In Figure 1b, point and area lights are presented according to the light source radius \( d \). The point light source was used to generate hard shadows, since there was only one path from a pixel to the light sample. In this case, \( N \) was one, and \( N_o \) was either zero or one in Equation (2). The area light produced a soft shadow where the intensity of the soft shadow was computed by the occlusion in Equation (2) or Equation (5). Note that generally smaller \( d \) generate more hard shadows. Moreover, we could simulate ambient occlusion by choosing a large \( d \) and small \( R \) and placing the light at \( \phi = \pi/2 \). This is illustrated in Figure 1c, which shows that we could produce similar conditions as in previous image-space ambient occlusion approaches, such as the image-space horizon-based ambient occlusion [5]. Figure 6 shows the different light sources applied to the engine block data. Only local illumination is used in (a), and our ISMOS model is applied in (b–d) as hard shadow, soft shadow, and ambient occlusion, respectively. (e) is a comparison image generated using the image-space horizon-based ambient occlusion [5].

![Figure 6](image-url)

**Figure 6.** Different light sources applied to the engine block data. Local illumination is applied in (a). Light intensity varies pixel-by-pixel in (b–d) due to the intensity attenuation. The image-space horizon-based ambient occlusion [5] is used in (e). Note that the local illumination indicates Phong shading.
3.5. Opacity Approximation

The partial occlusion computation described in Section 3.3.2 requires opacity values to accumulate the amount of light transported onto a pixel. In this section, we describe how to approximate opacity values continuously using volume transmittance.

Volume transmittance per pixel from an eye position during the ray-casting rendering was calculated as follows.

\[ T(z) = \exp\left(-\int_0^z \alpha(z')dz'\right) \quad (6) \]

\[ T_n \approx \prod_{j=1}^{n} \exp(-\alpha(z_j)\Delta z), \quad (7) \]

where \( n \) is the number of samples to reach the depth \( z \) during the ray-casting rendering, \( \Delta z \) the distance \( (z_j - z_{j-1}) \) between samples, and \( \alpha \) the opacity at a certain depth \( z \). The volume transmittance \( T(z) \) is a continuous analytical form, whereas \( T_n \) is a discrete form, which was used for the actual computation. The transmittance through objects is illustrated in Figure 2a and decreases as a ray passes an object. In previous work [34], a list of transmittances and their corresponding depths from a light position was stored for the deep shadow maps. To reduce the number of samples in the list, they introduced a compression scheme with an error tolerance, which is acceptable in practice. However, the compression procedure required a large storage for the list, and it might not be interactive when the light position changes. Instead, we proposed a functional approximation for the transmittance list. Figure 7a shows a graph of the transmittance as an example where the shape of the graph is similar to that of the Gaussian function. In practice, we do not need very accurate transmittance. The Gaussian function used in this work was defined as follows.

\[ f(z) = \exp(-\beta z^2), \quad (8) \]

where \( \beta \) is the parameter to be approximated. Note that we used \( f(z) \) to minimize the number of parameters that should be stored in a buffer, but other functional forms with more parameters can be used to approximate the transmittance more accurately. Since we wanted to find the best \( \beta \) to approximate \( T(z) \), we used least-squares fitting. The squared residual was defined as follows.

\[ R_{\text{residual}} = \sum_i \left(\beta z_i^2 - \log\left(\frac{1}{T_i}\right)\right)^2. \quad (9) \]

Figure 7. Transmittance approximation and multilayer. Actual volume transmittance (a) is computed using Equation (7). The approximate volume transmittance using a function is illustrated as the orange line in (b). The transmittance is computed per pixel. Both actual and approximate transmittance are computed in real time during the ray-casting rendering. Our ISMOS model is applied for transparent layers. (c) We used ISMOS only for the opaque layer (d3). (d) We computed the occlusions for all layers (d1, d2, d3).
From this, we could define \( X = (z_1, z_2, \ldots, z_n)^T \) and \( y = (\log(1/T_1), \log(1/T_2), \ldots, \log(1/T_n))^T \). Then, we simply applied linear least-squares fitting. Therefore, the best \( \beta \) per pixel was computed as:

\[
\beta = \left( X^T X \right)^{-1} X^T y. \tag{10}
\]

\( \beta \) was computed in real time during the ray-casting rendering. The approximate transmittance \( f(z) \) per pixel is shown in Figure 7b. Since we approximated the volume transmittance by only one parameter \( \beta \) per pixel, \( \beta \) was stored in the A component of the color buffer (RGBA) to represent the entire volume transmittance instead of using additional buffers to store opacity values corresponding to the depth layers. Note that the A component in the color buffer (RGBA) was not actually used as a color in our ISMOS model, since we did not use any blending after the first rendering pass; therefore, it was possible to use the A component for the approximation parameter \( \beta \) without using an additional buffer.

To calculate the approximate opacity value at a certain \( z \) from the transmittance approximation, we used Equations (11) and (12) in the following equations.

\[
f(z) = \frac{T_n}{T_{n-1}} = \exp(-\alpha(z)\Delta z). \tag{11}
\]

Therefore, the approximate opacity was computed as:

\[
\alpha_{app}(z) = \frac{1}{\Delta z} \cdot \log \left( \frac{f(z - \Delta z)}{f(z)} \right). \tag{12}
\]

This allowed us to easily and simply encode and decode the opacity values at any depth \( z \) position. As shown in Figure 2c, the opacity values were interactively calculated during the ray marching. We used this opacity approximation in Figure 5e with the partial occlusion computation. Note that extending our opacity approximation caused by the wavelength was possible by storing the wavelength information in an additional buffer.

### 3.6. Multiple Layers with Transparent Volume

We often render volumetric datasets with several transparent and opaque layers according to important features. In previous studies, the occlusion shading was applied to the entire volume. However, it was still difficult to see the shading effects on the transparent layers. Since our method operated in the image-space, applying our model to multiple layers was not easy with only one depth value per pixel. However, it was possible to find multiple layers depending on a layer differentiation method since we were using ray casting for rendering. In this work, we differentiated layers according to the transfer function settings. Note that depth peeling could be used to find multiple layers for polygonal rendering. Depth values for the layers were stored in depth buffers. The corresponding opacity values were calculated on-the-fly using our opacity approximation approach introduced in Section 3.5. We stored four front-most layers in the depth buffer as RGBA components, but it could be extended easily for more layers using more buffers. The multiple layers are illustrated in Figure 7c,d. In Figure 7c, we applied partial occlusion for pixels (d3 in the figure) on layers whose opacities (\( \alpha \)) were higher than \( \alpha_{th} \). For transparent pixels (d1 in the figure), we used local illumination only, such as Phong shading. The example shown in Figure 5f gave us an idea that our shading model was applied only on the opaque blue surface. We can see the shade through the transparent object (orange ball). The occlusion was computed using Equations (4) and (5).

However, applying our shading model only on opaque layers might produce unnatural effects. Instead, we computed occlusions on all layers and blended them. Figure 7d, for example, shows three layers (d1, d2, d3). d1 is not occluded, whereas d2 and d3 are partially and fully occluded, respectively. Therefore, \( O_{cc1} \) for d1 is 0, \( O_{cc2} \) is between 0 and 1, and \( O_{cc3} \) is 1. Once occlusions for all layers were calculated, we simply applied the alpha
blending technique for the final occlusion, $O_{cc}$ from the front layer (d1) to the back layer (d3). One disadvantage was that the partial occlusion computation had to be repeated for the number of layers, but the overhead was not significant.

3.7. Multiple Directional Light Sources and Spotlight

We introduced our image-space shading model with a single light source in the previous sections. Since we just needed to compute the occlusion factor $O_{cc}$ in Equations (2) and (5), it was straightforward to extend the idea to multiple directional light sources. We defined full occlusion in this work as:

$$O_{cc} = \left( \sum_j N_{oj} \right) / \left( \sum_j N_j \right),$$  \hspace{1cm} (13)

where $N_{oj}$ is the number of occluded samples from the $j$th light source and $N_j$ the total number of light samples on the $j$th light source. Similarly, partial occlusion was defined as:

$$O_{cc} = \frac{1}{\sum_j N_j} \left( \sum_i O_{cci} \right),$$ \hspace{1cm} (14)

We then blended the colors of all light sources and applied the occlusion similarly to Equation (3). Figure 8a presents an example with two light sources for the full occlusion case. The figure shows an example with two light sources $L_0$ and $L_1$. From a pixel $P$, there were four light samples on $L_0$ and five light samples on $L_1$. Three out of four samples were occluded for $L_0$, and two out of five were occluded for $L_1$. Then, we computed the occlusion from the two light sources as $O_{cc} = (3 + 2) / (4 + 5) = 0.555$. Let us assume that the colors for the two light sources are $C_0$ and $C_1$ and the intensities from the two lights are $I_{L0}$ and $I_{L1}$. The final color at the pixel $P$ was computed as:

$$C_{final} = \frac{1}{2} \cdot C_p \cdot (C_0 \cdot I_{L0} + C_1 \cdot I_{L1}) \cdot (1 - O_{cc}),$$  \hspace{1cm} (15)

where $C_p$ is the color at the pixel $P$ from the color buffer, and we ignored the $I_{amb}$ term for simplicity. Note that the number of light sources was not limited to two, although we used only two light sources for the demonstration. Figure 8 shows the effect of our multiple directional light sources using a teddy bear dataset. The ambient occlusion in (c) provides good depth cues compared to the local illumination in (b). However, it is still difficult to locate the arms in (c). The red and blue lights are applied separately in (d) and (e), and they provide more visual cues around the arms in each direction. (f) presents the effect of the two lights applied together, by which the visual cues are further enhanced. This multiple directional light effect could not be easily generated by previous image-space illumination models since the occlusion was computed at each pixel with light coming from all directions on the hemisphere. Note that the same lights used in (d) and (e) are applied in (f), and the two light colors are blended using Equation (15). Due to the light intensities, two different colors are blended gradually.

A spotlight can be used to emphasize a certain area. Since we had multidirectional occlusion settings, it was easy to model a spotlight. Figure 11e illustrates our spotlight model. The model was a cone with parameters, a spotting point ($C$) and a covering angle ($\omega$) from a light sample ($L_p$). We tested whether a pixel ($P$) was within a cone by comparing the angle $\gamma$ with $\omega$ (see Figure 11e for $\gamma$ and $\omega$). $\gamma$ was calculated as $\arccos\left(\frac{(P - L_p) \cdot (C - L_p)}{\|P - L_p\| \cdot \|C - L_p\|}\right)$. If $\gamma$ was smaller than $\omega$, then we computed the occlusions described in the previous sections. Otherwise, the pixel was occluded without further computations, and the color at the pixel was computed as follows.

$$C_{final} = C_p \cdot I_L \cdot I_{amb},$$ \hspace{1cm} (16)

where we assumed that the color of the light source was white.
Figure 8. Multiple directional light effect applied to the teddy bear data. The dual light model is presented in (a), but our system supports more than two light sources. The local illumination is applied in (b), and the image-space horizon-based ambient occlusion [5] is used in (c). (d,e) were obtained with single light sources and (f) shows the effect of multiple light sources.

4. Results and Discussion

We tested our approximate image-space multidirectional occlusion shading model on a Xeon CPU 2.66 GHz processor with NVidia GeForce GTX 295 graphics hardware, and for the volume rendering, we extended the ray-casting technique using the NVidia Cg shader. The color and depth from the volume rendering were stored in floating point textures, and they were used in our shading model, as described in Section 3. For the rendering without transparency, we used only one component in the depth texture; however, for the transparent object occlusion or multiple layer rendering, we used four components in the depth texture. As described in Section 3.5, the approximated opacity was easily computed using a parameter ($\beta$), which was stored together in the color buffer. Therefore, an additional opacity buffer was not required.

In the light model implementation presented in Section 3.2, the actual light samples were obtained by rotating samples on a disk with the radius $r$, whose center was at $(R + 0.5, 0.5, 0.0)$, to simplify the calculations. We rotated the samples by an angle $\phi$ about the y-axis ($R_y(\phi)$) and then rotated them once more by an angle $\theta$ about the z-axis ($R_z(\theta)$). The rotation matrix was obtained as follows.

$$R_z(\theta)R_y(\phi) = \begin{pmatrix}
\cos(\theta)\cos(\phi) & -\sin(\theta) & -\cos(\theta)\sin(\phi) \\
\sin(\theta)\cos(\phi) & \cos(\theta) & -\sin(\theta)\sin(\phi) \\
\sin(\phi) & 0 & \cos(\phi)
\end{pmatrix},$$

where $0 \leq \theta \leq 2\pi$ and $0 \leq \phi \leq \pi$.

We used six different datasets to show the effects of our ISMOS model, as summarized in Table 1. The engine data are presented with different shading methods in Figure 6. Our ISMOS model was able to produce the different shading effects by simply adjusting the light parameters. The shadow (b,c) and ambient occlusion (d) provided more depth cues compared to simple local illumination (a). The ambient occlusion effect (d) from our ISMOS was similar to the effect (e) proposed by Bavoil et al. [5]. Figure 8 presents the effect of the multiple directional light sources applied to the teddy bear dataset. The two directional lights in (f) provide more visual cues compared to the previous ambient occlusion method (c) and shading with a single light source (d,e).

We show an occlusion caused by transparent layers on an opaque layer in Figure 9. The transparent skin occludes the light arriving on the bone, and the effect is shown in (b) and (c). The occlusion is also caused by the bone, and this is seen around the back side of the neck bone. As seen in the figure, less light on the bone is observed in (c) compared to (b) since the more opaque skin in (c) occludes the light arriving at the bone more than the more transparent skin in (b). The occlusions on multiple layers are computed and blended in (d).
Table 1. Performance in fps. “1 L” means one light source, and “2 Ls” indicates two light sources. “FO” is full occlusion, and “PO” is partial occlusion. “PO, 1” indicates PO with one layer, and “PO, m” is PO with m layers. “\( \alpha_{app} \)” means that the opacity approximation is used in the column.

| Dataset     | Size          | Local Illumination | 1 L, FO | 1 L, PO, 1 | 1 L, PO, m | 1 L, PO, 1, \( \alpha_{app} \) | 1 L, PO, m, \( \alpha_{app} \) | 2 Ls, FO | 2 Ls, PO, 1 | 2 Ls, PO, m | 2 Ls, PO, 1, \( \alpha_{app} \) | 2 Ls, PO, m, \( \alpha_{app} \) |
|-------------|---------------|-------------------|--------|------------|-----------|-------------------------------|-------------------------------|--------|--------------|-------------|-------------------------------|-------------------------------|
| Engine      | 256 x 256 x 128 | 23.3              | 19.7   | 15.8       | 13.4      | 12.1                          | 11.2                          | 16.1   | 13.6         | 12.5        | 10.1                          | 9.2                           |
| Teddy bear  | 512 x 512 x 63  | 22.7              | 18.5   | 15.1       | 13.1      | 13.8                          | 12.5                          | 15.3   | 12.1         | 8.8         | 9.1                           | 8.6                           |
| Head        | 256 x 256 x 225 | 22.2              | 19.6   | 16.1       | 10.2      | 11.0                          | 8.3                           | 18.1   | 15.3         | 8.1         | 8.9                           | 7.0                           |
| Abdomen     | 256 x 256 x 128 | 20.0              | 13.5   | 11.2       | 10.8      | 8.9                           | 7.5                           | 11.4   | 9.7          | 8.6         | 7.9                           | 7.0                           |
| Carp        | 256 x 256 x 512 | 20.8              | 14.1   | 13.3       | 12.7      | 12.1                          | 11.8                          | 12.1   | 11.9         | 11.1        | 9.5                           | 8.8                           |
| Stag beetle | 832 x 832 x 494 | 10.4              | 9.2    | 8.2        | 7.9       | 7.5                           | 6.9                           | 7.8    | 6.4          | 6.0         | 5.5                           | 5.1                           |

Figure 9. Applying our ISMOS model for transparent scenes to the head dataset. (a) presents the direct volume rendering result with the local illumination. Our shading model is applied on the opaque bone layer in (b,c). More transparent skin is used for the partial occlusion in (b), whereas less transparent skin in (c). ISMOS is applied to all layers and the occlusions are blended in (d).

The abdomen data are compared with different shading methods in Figure 10. The data were rendered with the opaque transfer function setting in (a–c) and transparent setting in (d–f). (c) shows the full occlusion of ISMOS and provides very good depth cues compared to (a) and (b). We also compare the multiple layer effect in (e) and (f), and a part of the scene is highlighted with a blue box. The shading is visible through the transparent object, and the multilayer occlusion shows more natural shading effect as seen in (f). The carp data were used to show how well our ISMOS worked on a complicated structure in Figure 11. We extracted a bone structure out of the data and applied ISMOS on it. The opacity approximation is applied in (c,d), and we can see that the approximation is very reliable to represent the bone structure. Furthermore, the spotlight is used in (d), which enables us to focus on a certain area and to reduce the computation time since scenes outside the spotlight are simply occluded. Our ISMOS model in (c,d) provides proper visual cues of the object compared to (a,b). Similar effects are obtained with the beetle data as presented in Figure 12. We compare the accuracy of the opacity approximation in (c) and (d); and (f) and (g). Furthermore, we can see more natural shading effects with the multiple layer occlusion, and they are shown in (h) compared to (g). Moreover, the perceptual cue is improved with our ISMOS.

For the performance measurement, we used ray casting with a 580 x 580 view port in the volume rendering. The performances are summarized in Table 1. As shown in the table, the performance overhead was similar for all datasets. An average 13.8 ms overhead was measured for full occlusion with one light source and an average 25.6 ms overhead with two light sources. The average performance difference between the full and partial occlusion was about 11.4 ms, and the opacity approximation required on average 15.9 ms. The number of light sources affected the performance, but the difference was fairly small.
Note that our focus was not the optimization of the ray casting volume rendering, but the overhead for our shading model. Therefore, our performance on the volume rendering might not be faster compared to other highly optimized volume renderings, but the overhead was the same all the time as long as the view port size was fixed. Although there was a performance overhead by applying our ISMOS model, we still obtained interactive performance for all datasets even with transfer function changes and light source changes.

Figure 10. ISMOS applied to the abdomen data. Opaque and transparent transfer function settings are used in (a–c) and (d–f), respectively. The image-space horizon-based ambient occlusion [5] is used in (b). The approximated opacity values are applied for our ISMOS in (e–f). PO indicates partial occlusion.

Figure 11. ISMOS applied to the carp data. (a) presents the direct volume rendering with the local illumination. The approximated opacity values are used for our ISMOS in (c,d). The image-space horizon-based ambient occlusion [5] is used in (b). (e) presents our spotlight model.

For the quality comparison, we compare our hard shadow with the existing deep shadow map [27] in Figure 13. Local illumination in (a), ambient occlusion in (b), deep shadow map in (c), and our hard shadow by ISMOS in (d) are compared. As seen in the figure, our hard shadow is almost identical to the deep shadow map. Although our model supported various lighting models, indirect lighting was not achieved since indirect lighting was not interpreted within the model. However, it was possible to extend our model to support the enclosed volume with a partially opaque object and multiple objects with varying partial opacity by adding more buffers to store more precise transmittance.
Figure 12. ISMOS on the beetle data. Opaque and transparent transfer function settings are used in (a–d) and (e–h), respectively. The image-space horizon-based ambient occlusion [5] is used in (b).

Figure 13. Comparison of the deep shadow map [27] and ISMOS on the bonsai data. Local illumination in (a), ambient occlusion in (b), deep shadow map in (c), and our hard shadow by ISMOS in (d) are compared.

5. Conclusions and Future Work

In this work, we presented a novel approximate image-space multidirectional occlusion shading model for volume rendering. Our model was a unified shading model to simulate different light sources, and it provided more visual cues with minor performance overhead compared to using only local illumination. The occlusion for the opaque and transparent medium was handled separately using the full and partial occlusion methods. Our model caused small overhead in texture memory compared to the previous techniques and did not require any preprocessing. The transfer function and light position could be changed interactively. Moreover, we introduced an opacity approximation that allowed us to compute opacity values at any depth $z$ instead of using additional opacity buffers. In order to produce a more natural shading effect, we blended the occlusion shadings from multiple layers, and we were able to produce realistic shading results through a transparent volume. Multiple directional light sources enhanced visual cues more with a
small overhead. Then, we tested our ISMOS model with various datasets and proved that the ISMOS provided better visual cues with a small amount of additional computation. Our model can be applied to any rendering system since the ISMOS is decoupled from the rendering pass. As future work, we will optimize the ISMOS computation. Since the light source can be located anywhere in a volume, we will study an efficient image-space subsurface scattering effect. Moreover, we will investigate more efficient layer management to handle more layers in the future.

**Author Contributions:** All authors contributed to this study. Y.J. developed the system and wrote the article. S.K. developed a software module in the system. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the Basic Research Program through the National Research Foundation of Korea(NRF) funded by the MSIT (2019R1A4A1021702), in part by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korean government(MSIT) (No.2019-0-00242, Development of a Big Data Augmented Analysis Profiling Platform for Maximizing Reliability and Utilization of Big Data), and in part by the Institute for Information & communication Technology Planning & evaluation(IITP, Korea) funded by the Ministry of Science and ICT(MSIT, Korea) (2019-0-00795, Development of integrated cross-model data processing platform supporting a unified analysis of various big data models).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Jönsson, D.; Sundén, E.; Ynnerman, A.; Ropinski, T. A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering. *Comput. Graph. Forum* 2014, 33, 27–51. [CrossRef]

2. Schott, M.; Pegoraro, V.; Hansen, C.; Boulanger, K.; Bouatouch, K. A Directional Occlusion Shading Model for Interactive Direct Volume Rendering. *Comput. Graph. Forum* 2009, 28, 855–862. [CrossRef]

3. Šoltészová, V.; Patel, D.; Bruckner, S.; Viola, I. A Multidirectional Occlusion Shading Model for Direct Volume Rendering. *Comput. Graph. Forum* 2010, 29, 883–891. [CrossRef]

4. Bavoil, L.; Sainz, M. Screen Space Ambient Occlusion; Technical Report; NVIDIA Corporation: Santa Clara, CA, USA, 2008.

5. Bavoil, L.; Sainz, M.; Dimitrov, R. Image-space horizon-based ambient occlusion. In Proceedings of the ACM SIGGRAPH 2008 Talks, Los Angeles, CA, USA, 11–15 August 2008; p. 22:1.

6. Dimitrov, R.; Bavoil, L.; Sainz, M. Horizon-split ambient occlusion. In Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games, Redwood City, CA, USA, 15–17 February 2008; ACM: New York, NY, USA, 2008; p. 5-1.

7. Luft, T.; Colditz, C.; Deussen, O. Image Enhancement By Unsharp Masking The Depth Buffer. *ACM Trans. Graph.* 2006, 25, 1206–1213. [CrossRef]

8. Nichols, G.; Penmatsa, R.; Wyman, C. Interactive, Multiresolution Image-Space Rendering for Dynamic Area Lighting. *Comput. Graph. Forum* 2010, 29, 1279–1288. [CrossRef]

9. Ritschel, T.; Grosch, T.; Seidel, H.P. Approximating dynamic global illumination in image-space. In Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games, Boston, MA, USA, 27 February–1 March 2009; pp. 75–82.

10. Max, N. Optical Models for Direct Volume Rendering. *IEEE Trans. Vis. Comput. Graph.* 1995, 1, 99–108. [CrossRef]

11. Levoy, M. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.* 1988, 8, 29–37. [CrossRef]

12. Kniss, J.; Premoze, S.; Hansen, C.; Shirley, P.; McPherson, A. A Model for Volume Lighting and Modeling. *IEEE Trans. Vis. Comput. Graph.* 2003, 9, 150–162. [CrossRef]

13. Herrn, F.; Ljung, P.; Ynnerman, A. Interactive Global Light Propagation in Direct Volume Rendering using Local Piecewise Integration. In Proceedings of the IEEE/EG Symposium on Volume and Point-Based Graphics, Los Angeles, CA, USA, 10–11 August 2008; Hege, H.C., Laidlaw, D., Pajarola, R., Staadt, O., Eds.; The Eurographics Association: Aire-la-Ville, Switzerland, 2008. [CrossRef]

14. Tarini, M.; Cignoni, P.; Montani, C. Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization. *IEEE Trans. Vis. Comput. Graph.* 2006, 12, 1237–1244. [CrossRef]

15. Diaz, J.; Vázquez, P.P.; Navazo, I.; Duguet, F. Technical Section: Real-time ambient occlusion and halos with Summed Area Tables. *Comput. Graph.* 2010, 34, 337–350. [CrossRef]

16. Herrn, F.; Ljung, P.; Ynnerman, A. Local Ambient Occlusion in Direct Volume Rendering. *IEEE Trans. Vis. Comput. Graph.* 2010, 16, 548–559. [CrossRef]
17. Jönsson, D.; Kronander, J.; Ropinski, T.; Ynnerman, A. Historygrams: Enabling Interactive Global Illumination in Direct Volume Rendering using Photon Mapping. *IEEE Trans. Vis. Comput. Graph.* 2012, 18, 2364–2371. [CrossRef]

18. Kronander, J.; Jönsson, D.; Löw, J.; Ljung, P.; Ynnerman, A.; Unger, J. Efficient Visibility Encoding for Dynamic Illumination in Direct Volume Rendering. *IEEE Trans. Vis. Comput. Graph.* 2011, 18, 447–462. [CrossRef]

19. Lindemann, F.; Ropinski, T. Advanced Light Material Interaction for Direct Volume Rendering. In Proceedings of the IEEE/EG International Symposium on Volume Graphics, Norrköping, Sweden, 2–3 May 2010; Westermann, R., Kindlmann, G., Eds.; Eurographics Association: Aire-la-Ville, Switzerland, 2010; pp. 101–108.

20. Lindemann, F.; Ropinski, T. About the Influence of Illumination Models on Image Comprehension in Direct Volume Rendering. *IEEE Trans. Vis. Comput. Graph.* 2011, 17, 1922–1931. [CrossRef]

21. Ropinski, T.; Döring, C.; Rezk-Salama, C. Advanced Volume Illumination with Unconstrained Light Source Positioning. *IEEE Comput. Graph. Appl.* 2010, 30, 29–41. [CrossRef] [PubMed]

22. Ropinski, T.; Döring, C.; Rezk-Salama, C. Interactive Volumetric Lighting Simulating Scattering and Shadowing. In Proceedings of the IEEE Pacific Visualization Symposium (PacificVis 2010), Taipei, Taiwan, 2–5 March 2010; pp. 169–176.

23. Ropinski, T.; Meyer-Spradow, J.; Diepenbrock, S.; Mensmann, J.; Hinrichs, K.H. Interactive Volume Rendering with Dynamic Ambient Occlusion and Color Bleeding. *Comput. Graph. Forum Eurogr.* 2008, 27, 567–576. [CrossRef]

24. Sundén, E.; Ynnerman, A.; Ropinski, T. Image Plane Sweep Volume Illumination. *IEEE Trans. Vis. Comput. Graph.* 2011, 17, 2125–2134. [CrossRef] [PubMed]

25. Papaioannou, G.; Menexi, M.L.; Papadopoulos, C. Real-Time Volume-Based Ambient Occlusion. *IEEE Trans. Vis. Comput. Graph.* 2010, 16, 752–762. [CrossRef]

26. Tikhonova, A.; Correa, C.D.; Ma, K.L. Visualization by Proxy: A Novel Framework for Deferred Interaction with Volume Data. *IEEE Trans. Vis. Comput. Graph.* 2010, 16, 1551–1559. [CrossRef]

27. Hadwiger, M.; Kratz, A.; Sigg, C.; Bühler, K. GPU-accelerated deep shadow maps for direct volume rendering. In Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware, Vienna, Austria, 3–4 September 2006; ACM: New York, NY, USA, 2006; pp. 49–52. [CrossRef]

28. Annen, T.; Mertens, T.; Seidel, H.P.; Flerackers, E.; Kautz, J. Exponential Shadow Maps. In Proceedings of the Graphics Interface, Windsor, ON, Canada, 28–30 May 2008; pp. 155–161.

29. Banks, D.C.; Beason, K. Decoupling Illumination from Isosurface Generation Using 4D Light Transport. *IEEE Trans. Vis. Comput. Graph.* 2009, 15, 1595–1602. [CrossRef]

30. Ament, M.; Sadlo, F.; Weiskopf, D. Ambient Volume Scattering. *IEEE Trans. Vis. Comput. Graph.* 2013, 19, 2936–2945. [CrossRef] [PubMed]

31. Zhang, Y.; Ma, K.L. Lighting Design for Globally Illuminated Volume Rendering. *IEEE Trans. Vis. Comput. Graph.* 2013, 19, 2946–2955. [CrossRef] [PubMed]

32. Zhang, Y.; Ma, K.L. Fast Global Illumination for Interactive Volume Visualization. In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Orlando, FL, USA, 22–24 March 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 55–62. [CrossRef]

33. Timonen, V.; Westerholm, J. Scalable Height Field Self-Shadowing. *Comput. Graph. Forum Proc. Eurogr.* 2010, 29, 723–731. [CrossRef]

34. Lokovic, T.; Veach, E. Deep shadow maps. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LA, USA, 23–28 July 2000; pp. 385–392.