Fast Occlusion Sweeping

Mayank Singh[1⋆], Cem Yuksel[1⋆⋆], and Donald House[2⋆⋆⋆]

[1] Texas A&M University
[2] Clemson University

Abstract. While realistic illumination significantly improves the visual quality and perception of rendered images, it is often very expensive to compute. In this paper, we propose a new algorithm for embedding a global ambient occlusion computation within the fast sweeping algorithm while determining isosurfaces. With this method we can approximate ambient occlusion for rendering volumetric data with minimal additional cost over fast sweeping. We compare visualizations rendered with our algorithm to visualizations computed with only local shading, and with a ambient occlusion calculation using Monte Carlo sampling method. We also show how this method can be used for approximating low frequency shadows and subsurface scattering.

Realistic illumination techniques used in digitally synthesized images are known to greatly enhance the perception of shape. This is as true for renderings of volume data as it is for geometric models. For example, Qiu et al. [1] used full global illumination techniques to improve visualizations of volumetric data, and Stewart [2] shows how computation of local ambient occlusion enhances the perception of grooves in a brain CT scanned dataset. Tarini et al. [3] observed that perception of depth for large molecules was significantly improved with the use of ambient occlusion as compared to standard direct shading methods even when coupled with other techniques such as depth cueing and shadowing. Recently, a carefully designed experimental study by Weigle and Banks [4] definitively demonstrated that physically-based global illumination is a powerful adjunct to perspective projection in aiding human subjects to understand spatial relationships in a complex volume rendered scene. Despite the strong evidence for its efficacy in conveying spatial information in visualization, the use of global illumination is rare in practical visualization systems. This is most likely due to the high overhead of existing global illumination rendering algorithms.

In this paper, we provide a new solution for ambient occlusion computation that is significantly faster than existing techniques. The method integrates well with a volumetric ray marching algorithm implemented on the GPU. While not a full global illumination solution, ambient occlusion provides a more realistic illumination model than does local illumination, and permits the use of realistic light sources, like skylights. For accelerating our ray marching algorithm, we build a volumetric signed distance field using the *fast sweeping method*, and we embed our ambient occlusion approximation

⋆ e-mail: mayank@cs.tamu.edu
⋆⋆ e-mail: cem@cs.tamu.edu
⋆⋆⋆ e-mail: dhouse@cs.clemson.edu

directly into the sweeping algorithm. Thus, our algorithm can produce an ambient occlusion estimate with only a minor computational overhead. We are also able to use our approach to approximate low-frequency shadows due to direct illumination from certain angles, and to approximate subsurface scattering effects.

1 Background

Since our method combines an ambient occlusion computation with the fast sweeping method, in this section, we briefly overview the fast sweeping method, and ambient occlusion, and review previous work on computing ambient occlusion.

1.1 Fast Sweeping

The aim of fast sweeping is to build a volumetric signed distance field from volume data, for a specified isolevel. This defines a surface in 3D as the zero distance level set of the field, with all cells away from this surface containing the minimum distance from that cell to the surface. This distance field is commonly used as an aid in speeding the process of isosurface rendering using methods such as ray-marching. The fast sweeping method was introduced by Zhao [5] as a linear time alternative to the *fast marching* method [6] for computing a signed distance field.

The fast sweeping method can be divided into two distinct steps, as indicated in Fig. 1. First, the distance values of all grid vertices are initialized to positive infinity. Then, all vertices that participate in grid edges with exact zero-crossings of the isolevel (grey curve in the figure) are updated to their interpolated distance from the isolevel (open vertices in the figure).

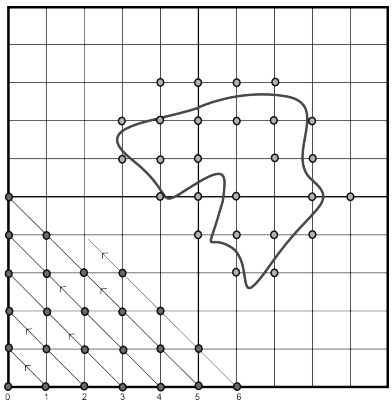


Fig. 1. The fast sweeping method for building a distance field to the zero level set (grey curve) in 2D. The open vertices indicate voxels initialized by direct distance computation. The closed vertices show the sweeping order of voxels for one diagonal sweep.

The second step conceptually consists of multiple diagonal sweeps that update the distance values from the distance values of neighboring vertices. In Fig. 1, the filled vertices show the order of vertices visited by one diagonal sweep, starting from bottom-left towards the top-right (the scanlines are numbered according to sweeping order).

This would be followed by a sweep back from the upper-right to the lower-left, and then sweeps in both directions along the other diagonal. 3D fast sweeping uses eight similar diagonal sweeps. In actual implementation, it is usual to arrange the algorithm so that iterations are across the columnar directions in the volume, instead of along diagonals.

During a sweeping operation in 3D, we consider the current distance field values $D(\mathbf{P})$ at three neighboring vertices of each vertex $\mathbf{P}$ in the grid. These three neighboring vertices are the ones that are visited before $\mathbf{P}$ depending on the current sweeping direction. Table 1 shows the sweeping directions and formulations of neighboring vertices $\mathbf{P}_s^x$, $\mathbf{P}_s^y$, and $\mathbf{P}_s^z$ for each sweep s, through a cubic grid whose spacing is h.

Table 1. Sweeping Directions and Neighboring Positions

s	Sweep Direction	$\mathbf{P}_s^x$	$\mathbf{P}_s^y$	$\mathbf{P}_s^z$
1	(1, 1, 1)	$\mathbf{P}-(h,0,0)$	$\mathbf{P}-(0,h,0)$	$\mathbf{P}-(0,0,h)$
2	(-1, 1, 1)	$\mathbf{P}+(h,0,0)$	$\mathbf{P}-(0,h,0)$	$\mathbf{P}-(0,0,h)$
3	(1,-1, 1)	$\mathbf{P}-(h,0,0)$	$\mathbf{P}+(0,h,0)$	$\mathbf{P}-(0,0,h)$
4	(-1,-1, 1)	$\mathbf{P}+(h,0,0)$	$\mathbf{P}+(0,h,0)$	$\mathbf{P}-(0,0,h)$
5	(1, 1,-1)	$\mathbf{P}-(h,0,0)$	$\mathbf{P}-(0,h,0)$	$\mathbf{P}+(0,0,h)$
6	(-1, 1,-1)	$\mathbf{P}+(h,0,0)$	$\mathbf{P}-(0,h,0)$	$\mathbf{P}+(0,0,h)$
7	(1,-1,-1)	$\mathbf{P}-(h,0,0)$	$\mathbf{P}+(0,h,0)$	$\mathbf{P}+(0,0,h)$
8	(-1,-1,-1)	$\mathbf{P}+(h,0,0)$	$\mathbf{P}+(0,h,0)$	$\mathbf{P}+(0,0,h)$

For each vertex, the distance field $D(\mathbf{P})$ is updated as the minimum of its current value and a new distance estimate $\hat{d}$, calculated from the existing values $D(\mathbf{P}_s^x)$, $D(\mathbf{P}_s^y)$, and $D(\mathbf{P}_s^z)$ of the indicated three neighbors. To calculate the new estimate, these three distances are sorted into ascending order, and renamed d_1, d_2, and d_3. Then

$$\hat{d} = \begin{cases} s_1 = a_1 + h & \text{if } |s_1| < a_2, \\ s_2 = \frac{a_1+a_2+\sqrt{2h^2-(a_1-a_2)^2}}{2} & \text{if } |s_2| < a_3, \\ s_3 = \frac{a_1+a_2+a_3+\sqrt{3h^2-(a_1-a_2)^2-(a_1-a_3)^2-(a_2-a_3)^2}}{3} & \text{otherwise.} \end{cases} \tag{1}$$

1.2 Ambient Occlusion

Ambient occlusion was first introduced by Zhukov et al. [7] as a better approximation to ambient light than the constant ambient term used in many local illumination models. Ambient occlusion on a point $\mathbf{P}$ with surface normal $\mathbf{N}$ is defined as

$$A(\mathbf{P},\mathbf{N}) = \int_{\Omega} V(\mathbf{P},\omega)\, g(\mathbf{N},\omega) d\omega\,, \tag{2}$$

where Ω is the unit sphere, $V(\mathbf{P},\omega)$ is the *visibility function* accounting for occlusion and scattering along direction ω, and $g(\mathbf{N},\omega) = \max(\mathbf{N}\cdot\omega, 0)$ is the *geometry term*. In this formulation, ambient occlusion corresponds to illumination due to an isotropic skylight including global shadows cast by surrounding objects (i.e. occlusion). It is the

computation of the visibility function for all directions that makes the ambient occlusion computationally intensive.

The typical way of solving Equation 2 is through Monte Carlo integration using raycasting for a binary decision of visibility in a chosen direction (if the ray hits any object visibility is zero in that direction, otherwise it is one). This approach requires many samples to reduce noise and it is very slow, especially when the integration is performed for all pixels of an image. Therefore, for real-time visualizations of static objects, ambient occlusion is often precomputed.

A popular alternative to Monte Carlo integration for ambient occlusion is precomputing shadows from multiple directional light sources (such as in [8, 9]). While this method can be faster, depending on the number of light directions and the complexity of the scene, it tends to produce undesired aliasing artifacts instead of noise.

For ambient occlusion in dynamic scenes, researchers have proposed different ways of keeping volumetric ambient occlusion fields around moving objects [10–13]. However, the initial computation of the ambient occlusion values is handled through a long precomputation step, similar to previous approaches.

For the purpose of ambient occlusion, Bunnell's [14] method represents each vertex in the scene as a planer disk called a *surface element*. The ambient occlusion computation is performed for every vertex by considering the contributions of all surface elements. Since the complexity of this algorithm is $O(n^2)$, where n is the number of surface elements, it is not suitable for scenes with high resolution surfaces.

Shanmugam and Arikan [15] used nearby pixels for computing local occlusion information. They precompute an expensive spherical representation of the model for distant occlusion information. Ritschel et al. [16] also used *only* local occlusion information derived from the nearby pixels in the image space. Their method does not account for global occlusion. Salama [17] proposed a multipass algorithm that precomputes a set of random directions and runs a GPU-based Monte Carlo raycasting.

2 Occlusion Sweeping

We compute an approximate ambient occlusion solution using eight sweeps of the fast sweeping method. During each diagonal sweep s, we compute the ambient contribution for each voxel from the corresponding octant of the sphere Ω_s. The final ambient occlusion is the sum of all of these eight octants, given by

$$A_s(\mathbf{P},\mathbf{N}) = \int_{\Omega_s} V(\mathbf{P},\boldsymbol{\omega})\, g(\mathbf{N},\boldsymbol{\omega})\, d\boldsymbol{\omega}\,. \tag{3}$$

Notice that the only difference between Equation 2 and Equation 3 is the integration domain.

To simplify the computation of this equation, we introduce three theoretical simplifications. First, we approximate the visibility function within an octant by a constant value $V_s(\mathbf{P})$. This implies that all of the light rays arriving at point $\mathbf{P}$ from the same octant are occluded the same amount. Using this simplification, we can take the visibility function outside of the integral and approximate ambient occlusion for the octant as

$$A_s(\mathbf{P},\mathbf{N}) \approx V_s(\mathbf{P})\, G_s(\mathbf{N})\,, \tag{4}$$

where $G_s(\mathbf{N})$ is the integrated geometry term

$$G_s(\mathbf{N}) = \int_{\Omega_s} g(\mathbf{N}, \omega)\, d\omega \,. \tag{5}$$

In this formulation $G_s(\mathbf{N})$ depends solely on the surface normal and can be easily precomputed. Therefore, all we need to compute to find the ambient occlusion is the visibility approximation of the octant for each voxel.

Our second simplification provides a fast estimation to $V_s(\mathbf{P})$ of the voxel at P by approximating it as a combination of the visibility at three neighboring voxels. These voxels, P_s^x, P_s^y, and P_s^z, are the three of the six neighbors of the voxel at P given in Table 1. We multiply the visibility values at these neighboring voxels by the transmittance $\tau(\mathbf{P})$ through these voxels to account for the light that is occluded within these voxels. As a result, our visibility estimation becomes

$$V_s(\mathbf{P}) \approx \frac{1}{3}\left[V_s(\mathbf{P}_s^x)\,\tau(\mathbf{P}_s^x) + V_s(\mathbf{P}_s^y)\,\tau(\mathbf{P}_s^y) + V_s(\mathbf{P}_s^z)\,\tau(\mathbf{P}_s^z)\right] . \tag{6}$$

Finally, we approximate the value of the transmittance function using the distance to the level set surface $D(\mathbf{P})$. We define the transmittance function to be

$$\tau(\mathbf{P}) = \begin{cases} 0 & \text{if } D(\mathbf{P}) \le -\frac{h}{2} \\ \frac{D(\mathbf{P})}{h} + \frac{1}{2} & \text{if } -\frac{h}{2} < D(\mathbf{P}) < \frac{h}{2} \\ 1 & \text{if } D(\mathbf{P}) \ge \frac{h}{2} \end{cases} . \tag{7}$$

To summarize, in our occlusion sweeping method, we visit each voxel eight times in eight diagonal sweeps. Each sweep computes the visibility function for the octant that is on the opposite side of the sweeping direction, using Equation 6. Table 1 gives the ordering of sweeping directions and the neighboring voxels used in the calculations. The order in which the voxels are visited in a particular sweep is chosen to assure that the visibility functions at the corresponding neighboring voxels of $\mathbf{P}$ are computed before $\mathbf{P}$. This whole computation can be easily implemented as a part of the fast sweeping method for building a signed distance field. Notice that even though we use the distance field values in Equation 7, the distance values within the range $[-\frac{h}{2}, \frac{h}{2}]$ are set during the initialization step. Therefore, we can use the distance field to evaluate $\tau(\mathbf{P})$ while building the distance field itself.

3 Implementation and Results

For demonstration purposes, we implemented our method within a single pass, GPU-based, ray marcher. The volumetric distance field and the ambient occlusion terms are precomputed offline, and stored as 3D textures on the GPU. A single ray is then marched for every pixel on the screen. If the ray hits a surface, hit position $\mathbf{P}$ is returned along with the gradient of the distance field, which is used as the surface normal $\mathbf{N}$. Since $\mathbf{N}$ is computed on the GPU per pixel (as opposed to per voxel of the dataset), Equation 4 needs to be evaluated separately for each octant at runtime. Therefore, we store the eight $V_s(\mathbf{P})$ fields as color channels in two 3D RGBA textures.

Since the eight integrated geometry terms $G_s(\mathbf{N})$ depend only on the surface normal, they can be precomputed once and stored in the color channels of two 2D textures (Fig. 2). The x and y coordinates of $\mathbf{N}$ then serve to determine the texture coordinate for lookup. These are shown in Fig. 2.

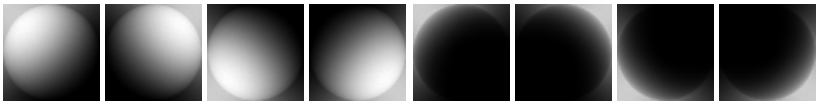
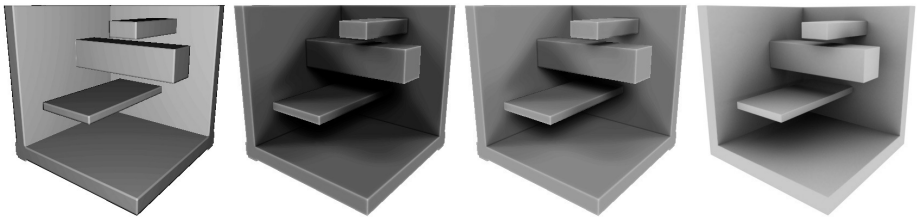


Fig. 2. Precomputed geometry term $G(\mathbf{N})$, stored as eight channels in two 2D RGBA textures.

Since ambient occlusion does not correctly account for inter-reflected light, as in a full global illumination calculation, images can be overly dark. To overcome the extra darkening, we applied gamma correction directly to our ambient occlusion estimation as a post-processing operation, computed as $A \leftarrow A(\mathbf{P},\mathbf{N})^{1/\gamma}$. We use $\gamma = 2$ for all gamma corrected images in this paper.



Local illumination **Occlusion Sweeping** **OcclusionSweeping** (gamma corrected) Monte Carlo Reference

Fig. 3. A simple $100 \times 100 \times 100$ dataset for demonstrating the validity of our algorithm, where we compare our occlusion sweeping results to local illumination and Monte Carlo reference.

To test the validity of our ambient occlusion estimation, we prepared the simple $100 \times 100 \times 100$ voxel dataset shown in Fig. 3, and rendered it using local illumination only, our occlusion sweeping method, and with a Monte Carlo ray tracer. The precomputation of ambient sweeping took only a fraction of a second, while the Monte Carlo reference image was rendered in several minutes. Both the occlusion sweeping and Monte Carlo images are arguably much more effective than the local illumination image in depicting the depth relationships of the various objects in the scene. It can also be seen that occlusion sweeping provides a good estimate for ambient occlusion, in that the dark areas and shadows correspond well to the those in the Monte Carlo reference image.

While occlusion sweeping greatly improves the illumination as compared to local illumination, there are two inaccuracies introduced by our simplifications. Ambient sweeping produces soft diagonal shadows, which are especially visible on the flat walls

of the images in Fig. 3. Secondly, ambient sweeping produces a glow effect along sharp edges. While this can be interpreted as a useful visualization feature (as argued by Tarini et al. [18]), it actually originates from our transmittance function $\tau(\mathbf{P})$ in Equation 7. Since $\tau(\mathbf{P})$ is directly computed from the distance field $D(\mathbf{P})$ and it does not depend on the octant, neighboring voxels on a flat surface end up partially occluding each other. While this effect darkens flat surfaces, sharp features are not affected as much and appear brighter.

Fig. 4, shows a comparison of local illumination to our ambient occlusion solution for a vorticity isosurface within a lightning cloud simulation dataset. As can be seen from this figure, ambient occlusion greatly improves depth perception and the global shape of this complicated surface becomes easier to understand. In Fig. 4, we provide a comparison of the gamma corrected occlusion sweeping to a Monte Carlo reference image for the same dataset. Notice that on a complicated surface like this one, the diagonal artifacts of ambient sweeping are not visible, while the darkening of flat regions emphasizes sharp local features.

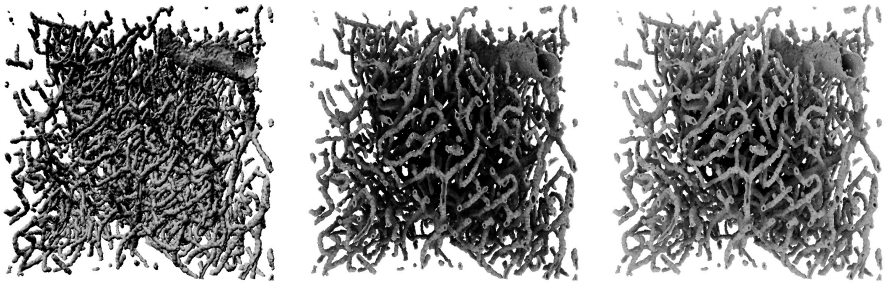
Local Illumination **Occlusion Sweeping** (gamma corrected) Monte Carlo Reference

Fig. 4. Comparison of occlusion sweeping to a Monte Carlo reference.

Fig. 5 shows another dataset that contains capillaries from the mouse brain (cerebellum). Notice that ambient occlusion with our method significantly improves the perception of the thread-like 3D vascular structure, while using only local illumination makes the image look flat and visually confusing.

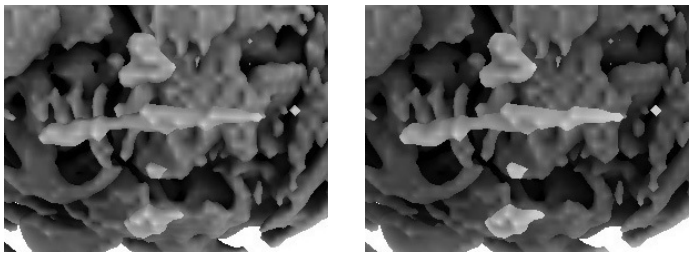
In our experiments, we noticed that local features can be further emphasized by dropping the geometry term $G_s(\mathbf{P})$ from Equation 4. In this case the visibility fields of all eight octants can be combined to form a scalar ambient occlusion field, significantly reducing the memory requirements. Fig. 6 shows such an example where it can be seen that sharp features appear brighter when the geometry term is dropped.

With our occlusion sweeping technique, we can also produce more natural illumination conditions than those produced by an isotropic skylight. For example, in Fig. 7 the surface is illuminated through only top four of the eight octants, emulating the lighting conditions of a cloudy day.



Local Illumination | **Occlusion Sweeping** | **Occlusion Sweeping** (gamma corrected)

Fig. 5. Vascular data of capillaries from the mouse brain (cerebellum) with different illumination methods. The dataset is about 0.5mm across and the capillaries occupy less than six percent of the overall tissue volume. The data is sampled on a $256 \times 256 \times 256$ grid.



Occlusion Sweeping | **Occlusion Sweeping** without G_s

Fig. 6. Ambient sweeping with and without the geometry term. Notice that the images are very similar. Eliminating the geometry term $G_s(\mathbf{N})$ makes edges and sharp features slightly brighter.

Table 2 provides timing information for our ambient occlusion precomputation, for five different data sets of varying sizes. The timings are computed on an Intel Core2 (2.66 GHz) machine with 4 GB RAM and NVidia GeForce 8800 GTX graphics card. Our method adds an overhead of from 30% to 100% to a standard fast sweeping calculation, which we believe to be bounded by memory access time, and not by computation. This is actually only a minor overhead to the full rendering calculation, as this computation needs to be done only once per viewing session.

4 Other Illumination Effects

4.1 Low-frequency Shadows

In our method, the result of ambient occlusion from each octant corresponds to low-frequency shadows from an area light source. The assumed directions of the light source is in the opposite direction to the sweeping direction. In this sense, our ambient occlusion solution can be thought of as the combination of low-frequency shadows from eight area light sources.

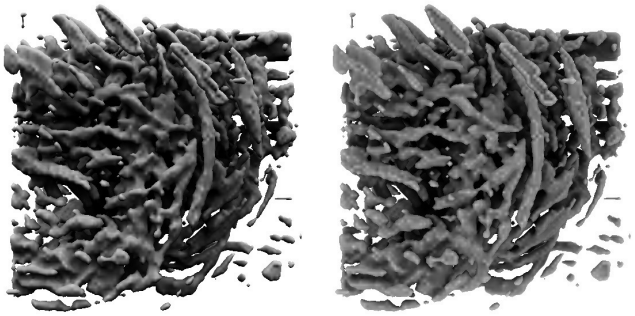


Fig. 7. Occlusion sweeping with lighting from only the top four octants, emulating the lighting conditions of a cloudy day. The image on the right is gamma corrected.

Table 2. Computational time (in seconds) for sweeping the dataset

NX NY NZ	Without Occlusion	Combined Occlusion	8 Octant Occlusion
101 101 61	0.72	0.91	1.01
100 100 100	1.06	1.386	1.52
150 150 150	3.86	6.52	7.78
200 200 200	9.56	17.12	23.43
250 250 250	23.26	39.66	46.64

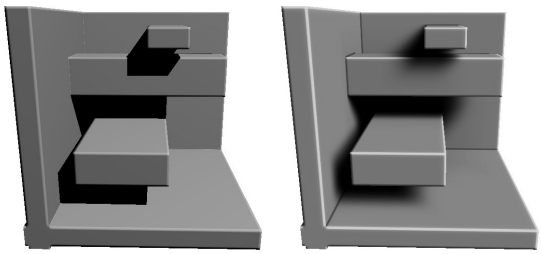
Fig. 8 shows a comparison of ray traced hard shadows to low-frequency shadows using a single octant of occlusion sweeping. While for simple datasets hard shadows can provide good depth cues, for more complicated datasets they are not as suitable. A complicated example comparing hard shadows to our low-frequency shadows are shown in Fig. 9. Notice that hard shadows of this complicated dataset produces additional high-frequency features, while our low-frequency shadows do not interfere with the actual data.

While the intensities of each of these eight light sources can be adjusted at run time, their positions are attached to the grid. For computing low frequency shadows with our technique from an arbitrary direction, one needs to rotate the grid on which the ambient occlusion is computed. While the ambient occlusion grid can be easily rotated independent of the distance field grid, changing the light direction requires recomputation of the occlusion values with a new sweep. However, the size of an area light source cannot be adjusted as it is a function of the grid resolution.

4.2 Subsurface Scattering Effects

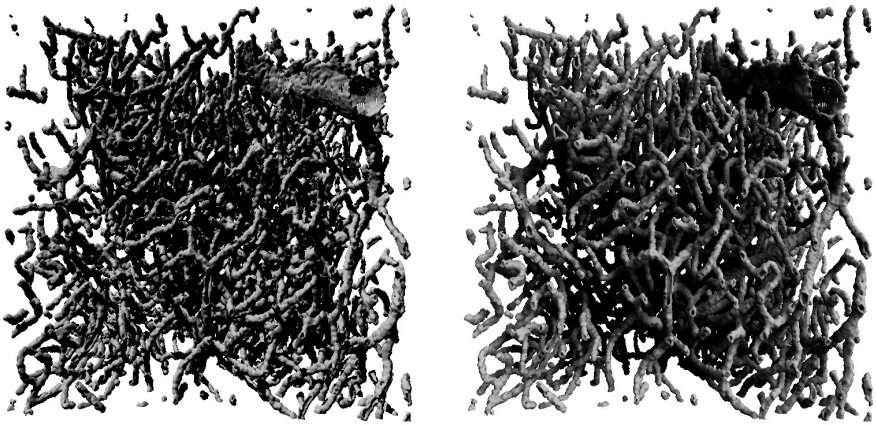
Subsurface scattering accounts for the light that enters a translucent surface and scatters within the object. For visualization purposes, subsurface scattering can be used to provide visual cues to the thickness of an object.

With a slight modification to our fast occlusion sweeping technique, we can achieve a subsurface scattering effect for materials with strong forward scattering. We perform



Ray traced shadows Low-frequency shadows

Fig. 8. Comparison of hard shadows computed using ray marching on the GPU to low-frequency shadows generated by our occlusion sweeping method.



Ray traced shadows Low-frequency shadows

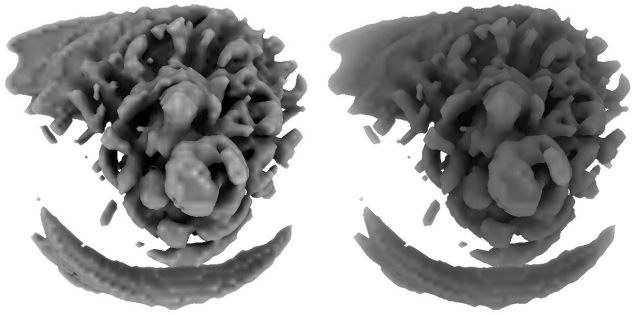
Fig. 9. Comparison of hard shadows computed using ray marching on the GPU to low-frequency shadows generated by our occlusion sweeping method for the vascular data in Fig. 5.

this by replacing the transmittance functions τ in Equation 6 with translucent transmittance functions τ_{trans}, such that

$$\tau_{trans}(\mathbf{P}) = 1 - (1 - \tau(\mathbf{P}))\ \alpha\ , \tag{8}$$

where α is a user defined opacity parameter between 0 and 1. Decreasing α values permit more light to penetrate through the surface, effectively softening the shadows of ambient occlusion and allowing surfaces to be lit from behind via forward scattering.

However, the geometry term $g(\mathbf{N}, \omega)$ in Equation 2 and the integral of the geometry term over an octant $G_s(\mathbf{N})$ eliminate the illumination contribution from the opposite direction of the surface normal. Therefore, for backward illumination from forward scattering, we need to modify the geometry term. We can easily achieve this by redefining it as $g(\mathbf{N}, \omega) = |\mathbf{N} \cdot \omega|$.



(a) $\alpha = 1$ (b) $\alpha = 0.6$

Fig. 10. With decreasing α values subsurface scattering effects become more prominent.

Fig. 10 shows examples of the subsurface scattering effect computed with our method. Notice that subsurface scattering makes thinner parts of the object brighter giving a visual cue indicating how deep the object is beyond the visible surface.

5 Conclusion

We have presented a new method, which we call *occlusion sweeping*, for the fast computation of ambient occlusion in the rendering of volumetric data. Unlike a full global illumination solution, our method produces approximate ambient occlusion in time that scales linearly with the size of the data set. Since it uses only volumetric calculations, its time complexity is not affected by the geometric complexity of the data. Further, it integrates very easily into the fast sweeping method for determining a signed distance field in a data volume. Therefore, our method should be particularly useful for visualization applications that use ray-marching through a distance field. In addition, our method permits other illumination effects like soft shadows and sub-surface scattering, both of which provide essential visual cues that enhance spatial perception.

As future work, we would like to explore other formulations for the visibility computation and experiment with more accurate transmittance functions to reduce or eliminate the illumination artifacts produced with our current algorithm.

6 Acknowledgements

This work was supported in part by the National Science Foundation ITR 0326194. We would like to thank David Mayerich for the mouse brain vascular data, and Edward Mansell for the lightning cloud simulation data.

References

1. Qiu, F., Xu, F., Fan, Z., Neophytos, N.: Lattice-based volumetric global illumination. IEEE Transactions on Visualization and Computer Graphics **13** (2007) 1576–1583 Fellow-Kaufman, Arie and Senior Member-Mueller, Klaus.
2. Stewart, A.J.: Vicinity shading for enhanced perception of volumetric data. In: IEEE Visualization. (2003)
3. Tarini, M., Cignoni, P., Montani, C.: Ambient occlusion and edge cueing for enhancing real time molecular visualization. Visualization and Computer Graphics, IEEE Transactions on **12** (2006) 1237–1244
4. Weigle, C., Banks, D.: A comparison of the perceptual benefits of linear perspective and physically-based illumination for display of dense 3d streamtubes. IEEE Transactions on Visualization and Computer Graphics **14** (2008) 1723–1730
5. Zhao, H.: A fast sweeping method for eikonal equations. Mathematics of Computation **74** (2005) 603–627
6. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. In: Proc. Nat. Acad. Sci. (1996) 1591–1595
7. Zhukov, S., Inoes, A., Kronin, G.: An ambient light illumination model. In Drettakis, G., Max, N., eds.: Rendering Techniques '98. Eurographics, Springer-Verlag Wien New York (1998) 45–56
8. Pharr, M., Green, S.: Ambient Occlusion. In: GPU Gems. Addison-Wesley (2004) 667–692
9. Sattler, M., Sarlette, R., Zachmann, G., Klein, R.: Hardware-accelerated ambient occlusion computation. In: 9th Int'l Fall Workshop VISION, MODELING, AND VISUALIZATION (VMV), Stanford (California), USA (2004) 119–135
10. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Trans. Graph. **21** (2002) 527–536
11. Mei, C., Shi, J., Wu, F.: Rendering with spherical radiance transport maps. Computer Graphics Forum **23** (2004) 281–290
12. Kontkanen, J., Laine, S.: Ambient occlusion fields. In: Proceedings of ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games, ACM Press (2005) 41–48
13. Malmer, M., Malmer, F., Assarsson, U., Holzschuch, N.: Fast precomputed ambient occlusion for proximity shadows. Journal of Graphics Tools **12** (2007) 59–71
14. Bunnell, M.: Dynamic Ambient Occlusion and Indirect Lighting. In: GPU Gems 2. Pearson Education, Inc. (2005)
15. Shanmugam, P., Arikan, O.: Hardware accelerated ambient occlusion techniques on gpus. In: I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games, New York, NY, USA, ACM (2007) 73–80
16. Ritschel, T., Grosch, T., Seidel, H.P.: Approximating dynamic global illumination in image space. In: I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games, New York, NY, USA, ACM (2009) 75–82
17. Salama, C.R.: Gpu-based monte-carlo volume raycasting. In: PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, Washington, DC, USA, IEEE Computer Society (2007) 411–414
18. Tarini, M., Cignoni, P., Montani, C.: Ambient occlusion and edge cueing to enhance real time molecular visualization (2006)