Fast GPU-based Visibility Computation for Natural Illumination of Volume Data Sets

Tobias Ritschel, Institute for Computational Visualistics, University Koblenz-Landau

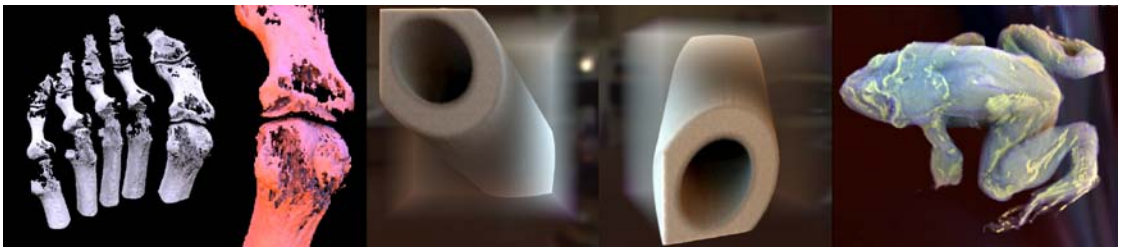

Figure 1: *After a few seconds of pre-calculation, volume data is rendered with natural illumination and soft shadows in realtime. The correct directional shadowing can be seen from the colored shadow cast inside the hollow tube using the kitchen light probe.*

Abstract

Pre-computed radiance transfer (PRT) has been used to render volumetric data under distant low-frequency illumination at real-time rates, including natural illumination, soft shadows, attenuation from semi-transparent occluders and multiple scattering. PRT requires a lengthy pre-process, which is acceptable only for static volume data. However, in practical volume rendering, general transfer functions are used. Manipulating such a transfer function will result in a dynamic radiance transfer which has to be re-computed. This work proposes a fast way for this re-computation. While previous work has used CPU Monte Carlo ray-tracing for pre-computation and requires time in the order of many minutes, our GPU implementation uses a hierarchical visibility approximation implemented entirely on the GPU and requires only a few seconds for typical scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism; I.3.3 [COMPUTER GRAPHICS]: Color, Shading, Shadowing and Texture

1. Introduction

Physically plausible lighting and shadowing for volume rendering is relevant for several reasons. First, shape perception, can be enhanced substantially by correct occlusion [Ste03]. Furthermore, the usage of natural lighting can enhance the perception of orientation of volumetric features. Shadows give additional visual cue of a feature's density: A dense feature will cast a more pronounced shadow. Besides that, interactive applications like games have an ever growing demand of visual realism that also includes volumetric effects like clouds, steam and smoke.

Rendering such effects at real-time rates has been demonstrated [SKS02] but required a lengthy pre-computation (PRT) that does not allow the volume or its transfer function to change. This work contributes a fast approach for this re-computation, because changing a volume is a typical operation in visualization: A user inspects a volume under high quality illumination at real-time rates by arbitrarily moving the camera, rotating the volume and changing the lighting. When the transfer function is changed, the radiance transfer is updated after a few seconds. This allows to use *progressive* PRT, where lighting details are added incrementally. Such a partial solution has no visual degradation compared to usual direct volume rendering – only the occlusion lacks details.

This paper is structured as follows: In Sec. 2 we review previous work and describe our approach in Sec. 3 for which results are presented in Sec. 4 before we conclude with Sec. 5.

2. Previous Work

The practical relevance of occlusion effects for shape perception is outlined by Stewart [Ste03]. Recent graphics hardware (GPUs) has made interactive high quality volume visualization possible [EHK*04]. Several researchers have included physically motivated illumination effects in volume rendering. Kniss et al. [KPHE02] simulate translucent volumes illuminated by a single directional light. Shadows in volumes from single directional light sources using deep shadow maps were demonstrated at interactive rates [HKSB06]. PRT [SKS02] has been used for low-frequency volume shadows. The work of Banks et al. [BB07] and Wyman et al. [WPSH06] demonstrate high quality volume rendering of iso-surfaces including soft shadows and indirect lighting. However, they require a lengthy pre-calculation and restrict lighting to be a function of the iso-value. We generalize this idea to arbitrary transfer functions, and handle iso-surfaces as a special case.

3. Our Approach

The emission-absorption volume rendering equation [Max95] for an isotropic medium is

$$L = \int_0^\infty E(\mathbf{x}(t))A(t)\mathrm{d}t$$

$$E(\mathbf{x}) = \int_\Omega \tau_\rho(D(\mathbf{x}))V(\mathbf{x},\omega)L_i(\omega)\mathrm{d}\omega$$

$$A(t) = \exp(-\int_0^t \tau_\alpha(D(\mathbf{x}(s)))\mathrm{d}s).$$

Here, $E(\mathbf{x})$ is the emission at location $\mathbf{x}$ and $A(t)$ is the absorption at t. The volume density at location $\mathbf{x}$ is denoted as $D(\mathbf{x})$. A transfer function τ maps a density value y to transparency $\tau_\alpha(y)$ and diffuse albedo $\tau_\rho(y)$. The incoming light $L_i(\omega)$ from direction ω is assumed to be independent of $\mathbf{x}$.

Central to our approach is the visibility $V(\mathbf{x},\omega)$ at location $\mathbf{x}$ in direction ω. We compute and store a full visibility function and not just a single average constant [Ste03] or a visibility for a single direction [KPHE02, HKSB06]. We denote this visibility field with V, where each voxel at location $\mathbf{x}$ stores visibility as a function of ω. Spherical harmonics (SH) have shown good results [SKS02, BB07, WPSH06] and will also be used to store an approximation of V denoted with $\bar{V}$ in this work. Therefore, two key components are required: One to compute $\bar{V}$ and one to visualize a density volume D under a given illumination $L_i(\omega)$ using $\bar{V}$. They are outlined in the following two sections.

3.1. Pre-Computaion

All pre-computation is done entirely using a GPU. To compute $\bar{V}$ from D, every voxel in V is traversed by drawing full-screen quads into all slices of a three-dimensional texture where one pixel corresponds to one voxel at location $\mathbf{x}$. The visibility function $V(\mathbf{x})$ is projected on the SH basis using Monte Carlo methods. This requires to evaluate V in N sample directions $\omega_1,\ldots,\omega_N$. To do this, we use a variant of DVR (Direct Volume Rendering) that simulates absorption only (no emission): A ray is started from $\mathbf{x}$ in direction ω_i and the absorption in D is evaluated front-to-back, including early ray termination. Repeating this with many uniformly distributed random samples gives the SH approximation $\bar{V}_l^m(\mathbf{x})$ of $V(\mathbf{x})$:

$$\bar{V}_l^m(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} V(\mathbf{x},\omega_i)Y_l^m(\omega_i), \qquad (1)$$

where $Y_l^m(\omega)$ is the SH basis l of some order m in direction ω. Finally $\bar{V}_l^m(\mathbf{x})$, which is a vector of SH-coefficients, is quantized to 8 bit / coefficient. Up to four coefficients per texture can be rendered, using multiple render targets. Using integer textures, current GPUs allow up to 16 coefficients, packed into four 32 bit integer color values. Typically 4 or 16 coefficients are used.

Hierarchical Visibility Using this representation allows to evaluate visibility in a certain direction from a certain position using a compact, pre-comptated SH field. However, computing $\bar{V}$ is costly for three reasons: The high spatial resolution of $\bar{V}$ (e.g. 256^3), the number of sample directions (e.g. $N = 500$) and the number of DVR steps (e.g. 256 steps) when computing $V(\mathbf{x},\omega_i),\ldots,V(\mathbf{x},\omega_N)$. Decreasing the resolution of $\bar{V}$ is straightforward but compromises shadow details (See Fig 4). A lower N will result in Monte Carlo noise, and miss angular details. Approximating $V(\mathbf{x},\omega_i)$ appears to be most suitable. However, simply using a low number of steps will not give a smooth approximation and ignore shadows of thin features entirely. We therefore make the simplifying assumption that regions which are distant to $\mathbf{x}$ can be approximated by a slow varying, low-resolution *expected opacity* volume O. This works because the projected solid angle for a feature of constant size decreases with the squared distance. Exactly locating and knowing the opacity variation across such a feature therefore becomes less important, most of all when using a band-limited SH representation for visibility anyway. This allows to traverse the volume with increasing step sizes and decreasing spatial resolution (See Fig. 2) in a logarithmic way when evaluating Eqn. 1.

We form a hierarchy of approximations $O_{1\ldots K}$, which is similar to a three-dimensional MIP map, but with a special minification filter. The default OpenGL way to compute lower MIP levels is suitable for reflectance in diffuse textures or irradiance [CB04]. However, using a linear filter (weighted sum) for expected opacity is not suitable. The ideal way to compute the expected opacity for voxel v is to average all possible path-integrals of absorption across v, which is intractable in practice. Instead, O_{i+1} is construct iteratively from O_i. First, the expected opacity value for each voxel in O_0 is computed. This is done by integrating absorption (applying τ_α to D)

across each voxel in the x, y and z direction, which is a mix of pre-classification and pre-integration [EHK*04]. Here, densities (before classification) are converted to opacities (after classification). This is done, because averaging densities does not make sense (See Fig. 2) with arbitrary transfer functions, while averaging opacities does.

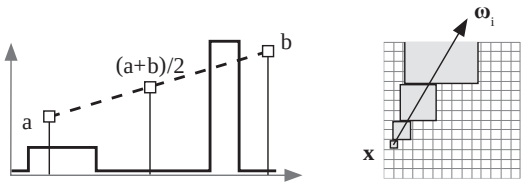

Figure 2: *Left: Averaging density a and b misses details from the transfer function. Right: Traversal with increasing step size.*

Higher levels are then computed as

$$D_{i+1}(\mathbf{x}) = \frac{1}{64} \sum_{j_1=0}^{1} \cdots \sum_{j_6=0}^{1} D_i(2\mathbf{x} + \begin{pmatrix} j_1 \\ j_2 \\ j_3 \end{pmatrix}) \cdot D_i(2\mathbf{x} + \begin{pmatrix} j_3 \\ j_4 \\ j_5 \end{pmatrix}).$$

This models the expected absorption as the traversal of all possible combinations of two successive voxels (See Fig. 3). As an example, consider a ray hitting an opaque voxel: The

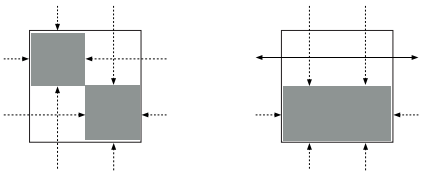

Figure 3: *A simplified minification example, assuming two-dimensions, binary opacities and discrete directions. Minification by box-filtering will result in 0.5 for both configurations. However, in the left configuration, more rays are blocked than for the right configuration. Our minifications accounts for this difference.*

ray is not expected to become *less* blocked from hitting another semi-opaque occluder. This is what averaging would result in, while multiplication behaves correctly. Note, that $O_{1\ldots K}$ is built only to accelerate the construction of $\bar{V}$ – it is not used in rendering at all. The minification is implemented on a GPU and takes only slightly longer then a standard MIP map creation. At runtime, $O_{1\ldots K}$ is sampled using the default trilinear mip-map filtering implemented in hardware, which gives the desired smoothness. The MIP level is selected manually proportional to the step size.

Sampling Implementation Details Monte Carlo computation with GPUs requires special attention as the random nature of Monte Carlo samples is a contradiction to the computational coherence required for good GPU performance. It has shown that jittered samples perform well when solving low-dimensional integration problems (where the curse of dimensionality is not a problem). This is, because the resulting memory and computation pattern is to some extent random and to some extent coherent at the same time: The i-th jittered sample direction will traverse the volume in roughly the same way. Besides that, the resulting integration quality is better. We therefore generate M jittered sample patterns with N samples each and store them into a two-dimensional texture. Each pattern stores N sample directions ω_i and the value $Y_l^m(\omega_i)$. For each $\mathbf{x}$ a different pattern is chosen randomly.

3.2. Runtime

Rendering means to generate a final image from D and $\bar{V}$ using distant lighting L_i. First L_i is also projected on the chosen SH basis as $\bar{L}_i$ for each color channel. Then, direct volume rendering (DVR) or direct iso-surface rendering (DIR) is used [EHK*04]. For DVR, emission and absorption is integrated along a view ray, requiring multiple evaluations of E. For DIR, the smallest t_0 such that $\tau_\alpha(D(\mathbf{x}(t_0))) = 0$ is found by ray-marching and a final binary search, requiring a single evaluation of E at t_0.

In both cases the emission $E(\mathbf{x})$ is not a constant anymore but depends on incoming radiance L_i, visibility $V(\mathbf{x})$ and albedo $\rho = \tau_\rho(D(\mathbf{x}))$. To evaluate E, we compute:

$$E(\mathbf{x}) = \int_\Omega \rho V(\mathbf{x}, \omega) L_i(\omega) \mathrm{d}\omega \approx \rho \bar{L}_i \cdot \bar{V}(\mathbf{x}).$$

In other words, the following steps have to be done in a fragment program:

- Load density $y = D(\mathbf{x})$
- Load albedo $\rho = \tau_\rho(y)$
- Load visibility $\bar{v} = \bar{V}(\mathbf{x})$
- Compute emission as a dot product of coefficents $\rho(\bar{v} \cdot \bar{L}_i)$

For a typical case (see Fig. 1) this requires one texture read for y, a texture read from the transfer function at y and four texture reads for $\bar{v}$ together with a dot product over a 16-vector in 4 instructions.

4. Results

The timings and quality achieved are demonstrated in Tbl. 1. The system used is an Intel Core 2 Duo 6300 and an Nvidia Geforce 8800 GTX. A sub-sampling ratio of $1:2$ or $1:4$ gives the best ratio between time and quality whereas full $1:1$ resolution is still to slow to be practical (See Fig. 4). The directional shadow quality can be seen in Fig 1: The shadow inside the hollow tube clearly reproduces the natural illumination from the kitchen environment map where red shadows

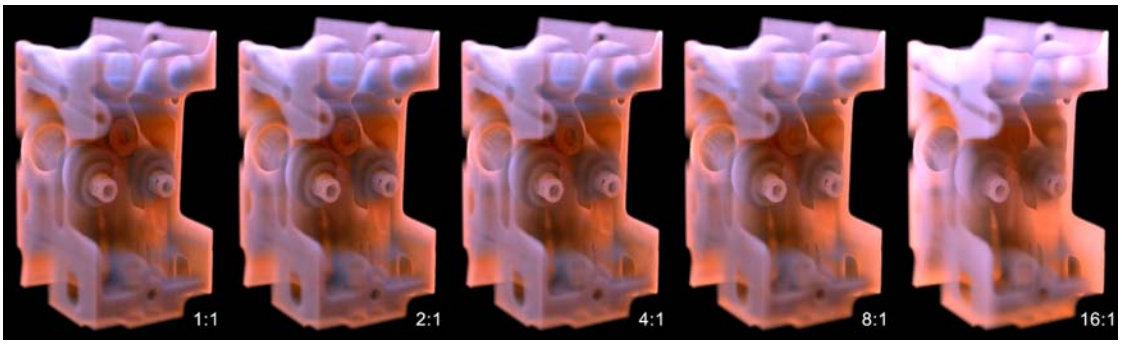

Figure 4: *Varying resolution of $\bar{V}$ for the engine block dataset. Renderings from a low-resolution $\bar{V}$ are slightly brighter because opacity can not increase as fast as in $\bar{V}$ at higher resolutions.*

appear when blue light is blocked and blue shadows can be seen when red light is blocked. The dynamic illumination effects can be seen in the accompanying video.

	Rate	Mem.	Setup (*s*)		Runtime (*ms*)	
			DVR	ISO	DVR	ISO
Engine	1 : 1	256 MB	70.4	40.8	71.2	17.2
	1 : 2	64 MB	12.3	7.6	45.6	15.7
	1 : 4	16 MB	4.0	2.6	40.2	12.6
	1 : 8	4 MB	2.3	1.5	36.1	9.5
Foot	1 : 1	256 MB	70.4	40.8	75.8	23.4
	1 : 2	64 MB	12.3	8.2	45.8	14.0
	1 : 4	16 MB	4.0	2.6	41.0	11.2
	1 : 8	4 MB	2.3	1.2	38.2	13.6

Table 1: *Timings for different scenes from Fig. 1 and Fig. 4. The iso-value used is* 0.8 *and* $N = 400$*. The screen resolution is* 1024×768*.*

5. Conclusion

We have demonstrated that our approach can render soft shadowing in volumes under varying natural illumination at real-time rates. When the volume is deformed or the transfer function has changed a pre-computation is carried out at near-interactive rates, which is an order of magnitude faster than previous CPU-variants, saves main memory, has better quality and supports arbitrary transfer functions at the same time.

For future work, we would like to use wavelets [NRH03] to allow more high frequency and view-dependent effects. The current use of SHs will not reproduce point-light-style sharp shadows – the same problem as encountered with SH surface shading. We would also like to experiment with more elaborate opacity or visibility approximations. The final objective however, is to have a fully dynamic volume with no pre-computation at all.

References

[BB07] BANKS D. C., BEASON K.: Fast Global Illumination for Visualizing Isosurfaces with a 3D Illumination Grid. *Computing in Science and Engineering 9*, 1 (2007), 48–54.

[CB04] CHRISTENSEN P. H., BATALI D.: An Irradiance Atlas for Global Illumination in Complex Production Scenes. In *Rendering Techniques* (2004), Keller A., Jensen H. W., (Eds.), Eurographics Association, pp. 133–142.

[EHK*04] ENGEL K., HADWIGER M., KNISS J. M., LEFOHN A., REZK-SALAMA C., WEISKOPF D.: Real-time volume graphics. In *Course 28 at ACM SIGGRAPH* (2004).

[HKSB06] HADWIGER M., KRATZ A., SIGG C., BÜHLER K.: GPU-Accelerated Deep Shadow Maps for Direct Volume Rendering. In *SIGGRAPH/Eurographics Graphics Hardware* (9 2006).

[KPHE02] KNISS J., PREMOZE S., HANSEN C., EBERT D.: Interactive Translucent Volume Rendering and Procedural Modeling. In *VIS '02: Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 109–116.

[Max95] MAX N.: Optical Models for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 99–108.

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22*, 3 (2003), 376–381.

[SKS02] SLOAN P.-P. J., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments. *ACM Trans. Graph. 21*, 3 (2002), 527–536.

[Ste03] STEWART A. J.: Vicinity Shading for Enhanced Perception of Volumetric Data. In *IEEE Visualization* (2003), Turk G., van Wijk J. J., II R. M., (Eds.), IEEE Computer Society, pp. 355–362.

[WPSH06] WYMAN C., PARKER S., SHIRLEY P., HANSEN C. D.: Interactive Display of Isosurfaces with Global Illumination. *IEEE Trans. Vis. Comput. Graph. 12*, 2 (2006), 186–196.