Interactive Volumetric Shadows in Participating Media with Single-Scattering

Chris Wyman*
University of Iowa

Shaun Ramsey†
Washington College

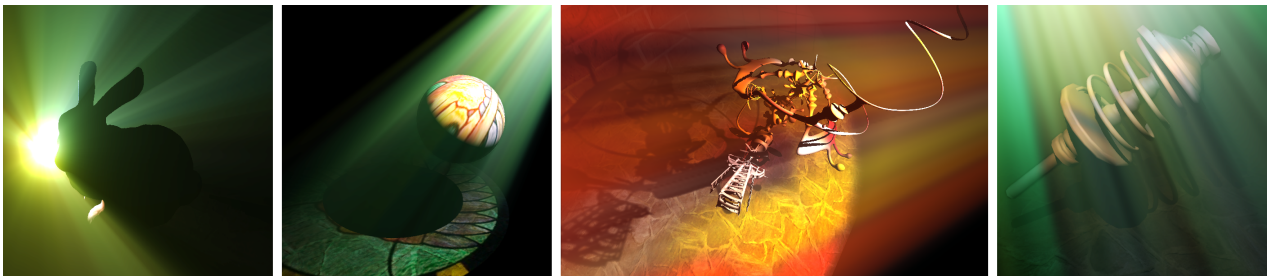

Figure 1: *Interactive volumetric shadows rendered at 40 to 80 fps with our hybrid shadow volume and ray marching technique.*

Abstract

Scattering effects arising from participating media, such as smoke, haze, and fog, dramatically add to perceived realism in renderings. As shadows affect illumination throughout an environment, they significantly diminish scattering effects in umbral regions. Unlike surface shadowing, accurate volumetric shadows require simultaneously integrating illumination, scattering, and attenuation throughout the volume, which proves challenging for interactive applications. We propose a method for rendering volumetric shadows in homogeneous single scattering media that combines ray marching and shadow volume techniques, eliminating performance deficiencies inherent in both. We extend this approach to interactively render shadows from textured lights and show results under two scattering models. Our prototype uses graphics hardware to accelerate shadow volume and ray marching computations. However since our hybrid selects sample points more intelligently than brute force techniques, it could also be applied to traditional ray tracing.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1 Introduction

Computer graphics applications make simplifying assumptions about scenes to maintain interactivity. One common assumption limits light interactions to surfaces, ignoring contributions from light scattered by particles in the air. Because haze, smoke, fog, and other precipitation are common in real environments, applications often fake their effects using simplistic distance-based models to approximately attenuate distant geometry. Recently introduced interactive methods [19] more realistically account for single scattering effects. However, these approaches ignore volumetric shadowing where occluders block light from scattering in umbral regions.

Volumetric effects have been well studied in the context of offline rendering. Typically renderers use ray marching methods to accumulate illumination, Monte Carlo techniques to probabilistically scatter particles, or volumetric photon maps to account for scattering in a two-pass process. These methods generally require extensive computation, restricting their use in real-time rendering.

Recently, researchers have proposed a number of methods for interactive volumetric shadows. Besides a few post-process techniques [16], these techniques broadly fall into two categories, based on either shadow volumes or ray marching. Methods using shadow volumes [3, 15, 10] identify shadowed segments of viewing rays that do not contribute to in-scattering. To correctly identify these segments, however, shadow polygons must be rendered back-to-front on a GPU, adding sorting costs to the rendering time.

Dobashi et al. [5] render quads parallel to the image plane at varying distances from the eye. These quads cut through a 3D representation of the participating media and are accumulated back-to-front using blending. Blending small values generally leads to precision errors; however, using fewer planes with larger individual contributions increases aliasing. Imagire et al. [9] address these problems by using fewer planes and averaging illumination over regions near each plane. Essentially, these techniques perform per-pixel ray marching through the volume, where sampling points in adjacent pixels are correlated by lying on planes perpendicular to the view.

We propose a hybrid method that avoids unpredictable numbers of depth peeling passes in shadow volume approaches by ray marching to identify contiguous segments of illumination along viewing rays. We reduce ray marching costs using analytic fog contributions [19] along illuminated ray segments, limiting the number of steps by only marching between front and back shadow polygons. Conceptually, this is similar to surface shadowing techniques that restrict work to necessary regions by culling [13] and stenciling [1, 4]. We also propose computing scattering at low resolution, allowing us to extend our work to scenes with textured spotlights for a modest cost increase. While this work currently aims to provide a method amenable to GPU implementation, future ray tracing implementations may ease standard GPU limitations, such as only using point light sources. For instance, the hybrid may be generalized to area light sources using penumbra wedges [2] instead of shadow volumes.

2 Participating Media Review

As light travels through participating media, particles modify the intensity through emission, absorption, in-scattering, and out-

*e-mail: cwyman@cs.uiowa.edu
†e-mail: sramsey2@washcoll.edu

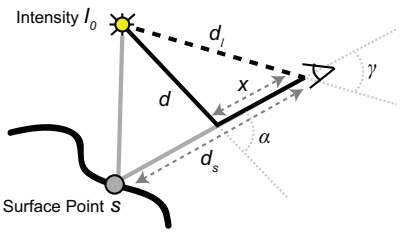

Figure 2: *Important paths of light in single-scattering participating media. d_s and d_l are the distances from the eye to the surface and light, x is the distance to a point on the viewing ray, and d is the distance from the light to this point.*

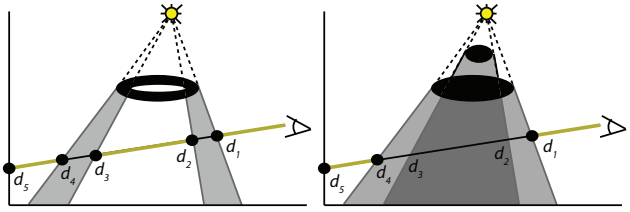

Figure 3: *(Left) Using a scattering model such as Sun et al.'s [19] airlight mode, fog contributions need only be computed at shadow polygons. (Right) Not every shadow polygon is important everywhere; rendering back-to-front allows identification of these cases, but at significant cost.*

scattering. As a full discussion of these phenomena is beyond the scope of this paper, we focus on a discussion of the homogeneous, single-scattering media assumed in our work. Interested readers can refer to standard texts (e.g., [7]) for more extensive coverage.

We use the following equation, based on the discussion by Nishita et al. [17], in our model:

$$\begin{aligned} L &= L_{attn} + L_{sctr} \\ &= L_s e^{-(\kappa_a+\kappa_s)d_s} + \int_0^{d_s} \kappa_s \rho(\alpha) \frac{I_0}{d^2} e^{-(\kappa_a+\kappa_s)(d+x)}\, dx, \end{aligned} \quad (1)$$

where the first term corresponds to the attenuated light reflected from surface point s and the second term represents the additional light contributed by in-scattering along the viewing ray. L_s represents the radiance reflected from s towards the eye, I_0 the light's emitted radiance, d_s the distance from the eye to s, and d the distance from the light to an arbitrary point on the viewing ray at distance x from the eye, as illustrated in Figure 2. κ_a and κ_s are the media's absorption and scattering coefficients, and $\rho(\alpha)$ is its phase function.

Obviously, the integration proves challenging in interactive contexts. Solutions are possible using numerically preintegrated values for each scene stored in tabular form, but more recent analytical approximations have produced good results. Of particular interest, Sun et al. [19] reduce this integral to two lookups into the function $F(u,v) = \int_0^v exp[-u\tan\xi]\, d\xi$:

$$L_{sctr} = I_0 A_0 \left[F\left(A_1, A_2\right) - F\left(A_1, \frac{\gamma}{2}\right) \right], \quad (2)$$

where $F(u,v)$ is precomputed and stored in a texture. A_i are functions of γ, d_s, d_l, and κ_s introduced for clarity:

$$A_0(d_l, \gamma, \kappa_s) = \frac{\kappa_s e^{-\kappa_s d_l \cos\gamma}}{2\pi d_l \sin\gamma}, \quad (3)$$

$$A_1(d_l, \gamma, \kappa_s) = \kappa_s d_l \sin\gamma, \quad (4)$$

$$A_2(d_l, \gamma, d_s) = \frac{\pi}{4} + \frac{1}{2}\arctan\frac{d_s - d_l\cos\gamma}{d_l \sin\gamma}, \quad (5)$$

Note that d_l, the distance from eye to the light, is constant per frame and κ_s is constant in homogeneous media. Thus, for any particular frame L_{sctr} varies only in d_s and γ, the angle between the view ray and the vector from eye to light. The supplementary material provides additional details about this model.

3 Volumetric Shadows

Various researchers [3, 15] have used shadow volumes to render volumetric shadows. The key idea is that regions between shadow quads have $I_0 = 0$ and thus do not contribute to the integral in Equation 1. This allows splitting the integral into multiple integrals, one for each illuminated interval. For the case in Figure 3:

$$L_{sctr} = \kappa_s I_0 \left[\int_0^{d_1} f(x)\, dx + \int_{d_2}^{d_3} f(x)\, dx + \int_{d_4}^{d_5} f(x)\, dx \right], \quad (6)$$

where $f(x) = \frac{\rho(\alpha)}{d^2} e^{-(\kappa_a+\kappa_s)(d+x)}$. Repeating Sun et al.'s [19] math, this simplifies to two lookups into the $F(u,v)$ texture for every illuminated segment along the viewing ray, where front facing polygons are positive contributions and back facing polygons are negative contributions.

However, not all shadow polygons contribute to this computation. For complex models, silhouette edges not visible from the light are extruded into extraneous polygons. Other shadow polygons are valid in only certain regions (as in Figure 3). Previously, solving these problems required repeated depth peeling to sort shadow quads [3]. This is costly for two reasons: numerous passes may be needed for even simple scenes and rendering shadow quads consumes significant fill rate even for single-pass stencil shadows.

Other approaches use volume rendering techniques, accumulating contributions along slices through the scene [5] or ray marching along each pixel [12]. These methods typically avoid aliasing by adding additional slices or ray steps. While this additional work is necessary near shadow boundaries, it is wasteful elsewhere.

3.1 Hybrid Ray Marching / Shadow Volumes

We propose a hybrid method that eliminates the sorting requirement of shadow volume approaches by using ray marching, but only marches in regions where shadow volumes identify that shadows exist. The basic idea can be described as follows:

1. Render a shadow map from the light;
2. Render shadow volumes, as seen from the eye, storing the distance to the frontmost and backmost polygons;
3. Render from the eye, attenuated by $e^{-(\kappa_\alpha+\kappa_s)d_s}$;
4. Use the airlight model [19] directly for eye pixels encountering no shadow quads;
5. Ray march from the back to front shadow quads, use the shadow map to identify illuminated regions (such as the segment d_2d_3 in Figure 3), use the airlight model (Equation 2) for each lit interval;
6. Combine attenuated scene and airlight contributions.

We perform this algorithm in four passes, combining the last three steps by drawing a full-screen quad, evaluating the airlight model (step 4) for fragments without a corresponding shadow volume, performing ray marching (step 5) for other pixels, and computing the sum from Equation 1 (step 6) for all pixels.

Step 2 can be done in a single pass by outputting depth (z) for front-facing polygons and an inverted depth ($1-z$) for back-facing polygons, either using GL_MIN blending to accumulate back and front contributions into different color channels or using a geometry shader to output front and back facing polygons into different buffers.

Since many scenes include volumetric shadows in only limited regions, this hybrid avoids marching completely in pixels unaffected by shadows. Additionally, since shadows often interact with

the media for a limited distance along any ray, this approach focuses samples in relevant regions, reducing the number of samples required to avoid aliasing. Finally, because ray marching starts and stops at shadow polygons, we guarantee that the frontmost and final shadow boundaries along each ray will not exhibit aliasing.

3.2 Adding Textured Spotlights

While the above approach quickly adds volumetric shadows to scenes with single color point lights, most scenes contain more interesting illumination.

For single-colored spotlights, one could simply use the above method, limiting computation to regions inside the spotlight (instead of from the eye to point s). However this approach does not work for multi-colored lights, for instance the illumination from a stained glass window. In these cases, additional stepping along the ray is necessary; as I_0 is no longer constant over lit regions, it cannot be pulled out of the integrals in Equation 6.

However, we can restrict ray marching to regions inside the light frustum. Furthermore, inside this frustum shadowed and illuminated regions can be sampled at different rates, so we use both the light frustum and shadow volumes to reduce marching costs. In this case, the algorithm becomes:

1. Render a shadow map;
2. Render shadow volumes, as seen from the eye;
3. Render from the eye, attenuated by $e^{-(\kappa_\alpha+\kappa_s)d_s}$;
4. Render light frustum from the eye, storing front and back distances;
5. Outside the light frustum, set scattering to 0;
6. Inside the light frustum, march from the back to front for pixels with no shadow volumes, sampling I_0 and scattering at each step;
7. For pixels encountering shadow volumes, ray march along three segments inside the light frustum: before the shadow, between shadow volumes, and after the shadow, sampling I_0 and scattering for each illuminated segment;
8. Combine attenuated scene and scattering contributions.

While one would initially think shadowed ray segments could use a coarser sampling, interestingly we often found that due to objectionable aliasing at shadow boundaries the segment between shadow quads needs denser sampling. And as discussed in the next section, due to the lower frequencies in illuminated participating media we used coarser sampling in illuminated regions.

3.3 Further Optimizations

Single-scattering models, by definition, only scatter once inside the participating media. In relatively thick fog or haze where volumetric shadows are most apparent, light often scatters repeatedly. In real environments, this means shafts of light or shadow lose the crisp, high-frequency edges seen under single scattering models.

This blurring allows three further optimizations to improve performance. First, we render scattering contributions at a much lower resolution than the final rendering. This reduces ray marching costs and rendering shadow polygons consumes less fill-rate. We found using a 256^2 in-scattering image was sufficient for a 1024^2 final rendering, though we apply a 3×3 Gaussian blur to the low resolution image to eliminate aliasing of very thin volumetric shadows.

Rendering low resolution scattering and upscaling in the final render requires care at depth discontinuities, as geometric discontinuities coinciding with high contrast changes in the airlight contribution appear aliased. This can be avoided by selecting the correct scattering contribution (in the final step) based upon fragment depth and not blurring the low resolution image over depth discontinuities. We essentially modify our 3×3 Gaussian into a bilateral filter [20], similar to a a number of existing techniques (e.g., [18]), though a variety of other noise reduction and antialiasing techniques could also work.

Using textured spotlights significantly increases aliasing potential, as coarse steps along the ray undersample high frequencies in the texture. Our second optimization arises from observing that these high frequencies are rarely visible in real environments, as multiple scattering diffuses sharp illumination edges. We use a downsampled, blurred image of the spotlight to reduce high frequency details. In some sense, the idea is similar to Sun et al.'s [19] precomputation of scattering from environment map illumination, though in our case it gives a precomputed *ad hoc* approximation of blurring from additional scattering events that simultaneously allows use to sample the volume's illumination less frequently.

Extending this idea further, a mipmap hierarchy for the spotlight texture can also reduce illumination aliasing by allowing determination of each ray interval's illumination, I_0, using only appropriate frequencies. This provides a tunable quality versus speed tradeoff, where rays outside the shadow volumes but inside the light frustum can be sampled arbitrarily coarsely in exchange for correspondingly blurry illumination from scattering.

Finally, the low frequency of volumetric scattering means the exact shadow boundaries are not so important. This allows shadow volumes from coarse occluders to stand in for those of full resolution geometry. Taken to the extreme, one could use an axis-aligned bounding box to approximate the occluder. However, we found that in order to maintain an alias-free shadow boundary, the low resolution model must have silhouettes that at least coarsely approximate those of the original. When using an axis-aligned bounding box, its frontmost "shadow volume" does not mark the start of the shadow region; the shadow map must be sampled to identify this boundary, which introduces shadow aliasing.

3.4 Implementation Details

A number of implementation tricks help us achieve real-time performance. For the airlight model (Equation 2), repeatedly computing the A_i terms each pixel for many ray steps is costly. Because the A_i terms vary only in two dimensions (d_s and γ) every frame, we can re-factor Equations 2 and 6:

$$L_{sctr} = \sum I_{span} \left[F'(\cos\gamma, x_{end}) - F'(\cos\gamma, x_{begin}) \right],$$

by rendering F' into a 2D texture once per frame, removing computations for the A_i terms from the inner ray marching loop. Here I_{span} is the intensity over a ray step and x_{begin} and x_{end} are distances from the eye to the beginning and end of the interval. F' is indexed by $\cos\gamma$ and distance x:

$$F'(\cos\gamma, x) = A_0(d_l, \gamma, \kappa_s) F\left(A_1(d_l, \gamma, \kappa_s), A_2(d_l, \gamma, x)\right),$$

which allows computation of each summation step with one texture lookup, one subtraction, and one multiply. See the supplementary materials for additional details.

Another improvement suggested by Imagire et al. [9] uses variance shadow maps [6] to return the probability a point is shadowed, rather than a binary visibility. This helps remove aliasing at shadow boundaries, though we found the additional cost to create the VSM is not worthwhile for all scenes—simply using smaller ray steps was sometimes as cost effective.

Finally, the key to reducing the number of ray marching samples lies in intelligently picking them. Using samples on image-parallel planes or of uniform sizes across all pixels leads to visible coherency artifacts in the image, similar to regular sampling artifacts during multisampling. We found uniformly sampling the distance between front and back shadow volumes worked well; while adjacent pixels maintain coherency, it is along non image-aligned surfaces that generally parallel shadow boundaries (see Figure 4). Currently, we determine the sampling rate empirically.

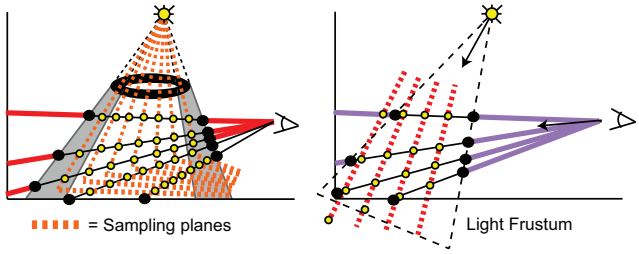

Figure 4: *(Left) When marching between shadow polygons, we sample the front-to-back distance uniformly per ray. (Right) When sampling light colors, bends in the sampling planes produce clearly visible artifacts. Instead we sample uniformly, with step size determined based on the angle between spotlight and each pixel's ray directions.*

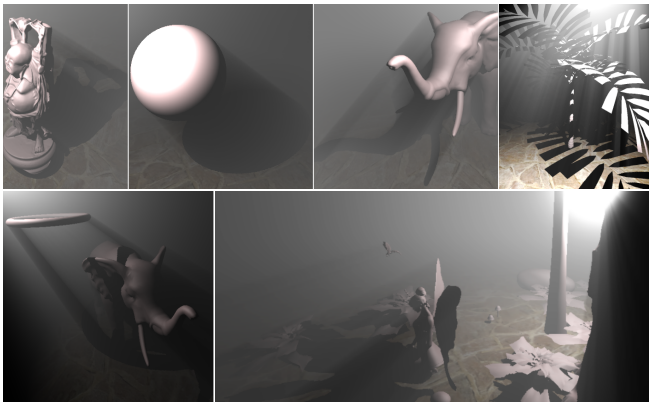

Figure 7: *Examples from scenes with varying object and depth complexities. Note that fairy scene objects not in the shadow map do not cast volumetric shadows, as the shadow map determines if each ray segment is illuminated and we chose to light all segments outside the light frustum.*

For textured spotlights, we initially uniformly subdivided this interval as well. However this adds artifacts at view frustum boundaries, where the sampling plane changes sharply. Instead we choose our sampling distance based on the angle between the spotlight direction and the ray direction, similar in spirit to the volume rendering sampling suggested by Kniss et al. [11]. See the paper's supplementary appendices for specifics on how we sample the ray.

4 Results

Our implementation runs in OpenGL using an nVidia GeForce 8800 GTX on a 2.66 GHz multi-core Xeon processor; the timings shown in Table 1 are for 1024^2 images. In order to allow a meaningful comparison between different scenes we use consistent sampling rates in these timings. For brute force ray marching we use 150 samples, with our hybrid we use 50 samples inside the light's shadow volumes, and for textured lights we take up to 150 ray steps inside the light frustum, with up to 50 each in front of, between, and behind the shadow volumes. These numbers were chosen to eliminate aliasing in most of our examples, though in some scenes (e.g., the sphere) such high sampling rates are unnecessary and in others (e.g., the YeahRight scene) higher sampling is necessary. Figures 5 and 6 examine the effect of sampling rates in our results and compare our work to equivalent cost and quality ray marching.

Figures 1 and 7 show additional examples of our volumetric shadows under both point lights and textured spotlights. Scenes illuminated by white point lights use Sun et al.'s [19] model while colored light scenes use Hoffman and Preetham's [8] model, which gave us better contrast in those scenes.

Using brute force ray marching, the computation costs depend

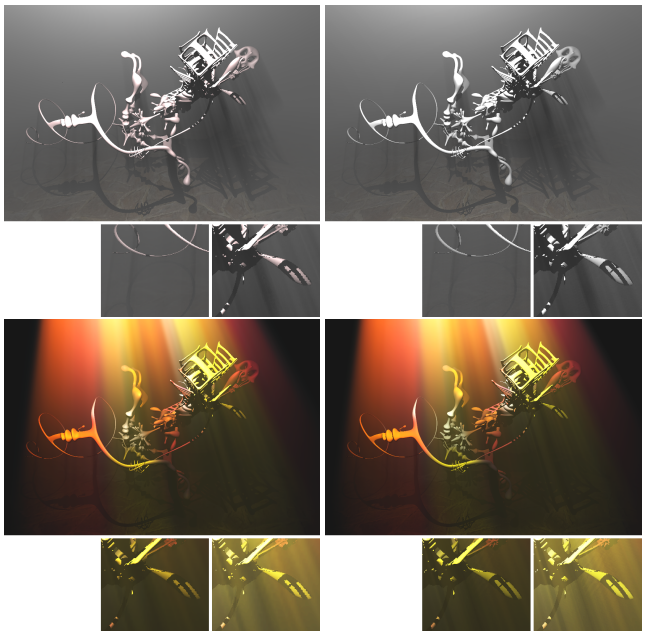

Figure 8: *(Left) Our hybrid with a point light using Sun et al.'s model and a textured spotlight using Hoffman and Preetham's model. (Right) Ray traced comparison, sampling scattering 500 times per pixel. The colored light insets show the same region, one with enhanced contrast.*

mainly on image resolution and number of steps, particularly when sampling the scattering at all 1024^2 pixels. Using our hybrid to selectively march behaves similarly when sampling at each pixel and the shadow or light frustums cover most of the scene (e.g. the spring and yeah right scenes). In the case of single-color point lights our framerates are mostly dependent on model complexity, and this dependency also emerges when computing low resolution scattering under textured lighting.

Limiting ray stepping to only between shadow polygons gives a 3–8× speedup over brute force techniques at 1024^2, depending on shadow size and volume, and computing scattering at lower resolutions adds another 25–100%. Performance with textured lights typically improves by a factor of 3–6× using lower resolutions, depending on if higher resolution costs are bound more by ray marching or geometry. Figure 6 shows that with increased samples along each ray marching, the hybrid has an increasingly larger performance advantage as shadow volume costs remain fixed but the savings from eliminated samples grows.

Figure 8 provides a ground truth comparison with an offline ray tracing using the same scattering models. As a point of reference, the ray traced results required 30 minutes per frame (but the ray tracer was completely unoptimized). The color insets show some remaining jaggies and halo artifacts around depth discontinuities caused by computing in-scattering at $\frac{1}{16}$ resolution, as well as aliasing caused by our use of shadow mapping for surface shadows.

The accompanying videos show that under animation our work is quite compelling, even with scattering computed at low resolution. A 3×3 bilateral filter suffices to eliminate most low-resolution jaggies from the in-scattering, even under animation, though some very thin shadows are missed completely or blurred away.

5 Discussion

A number of difficult cases exist for our hybrid. For scenes where shadows cover most of the scene (e.g., indoor scenes lit by sunlight through a window), shadow volume computations may become prohibitively expensive. One possibility would invert the acceleration,

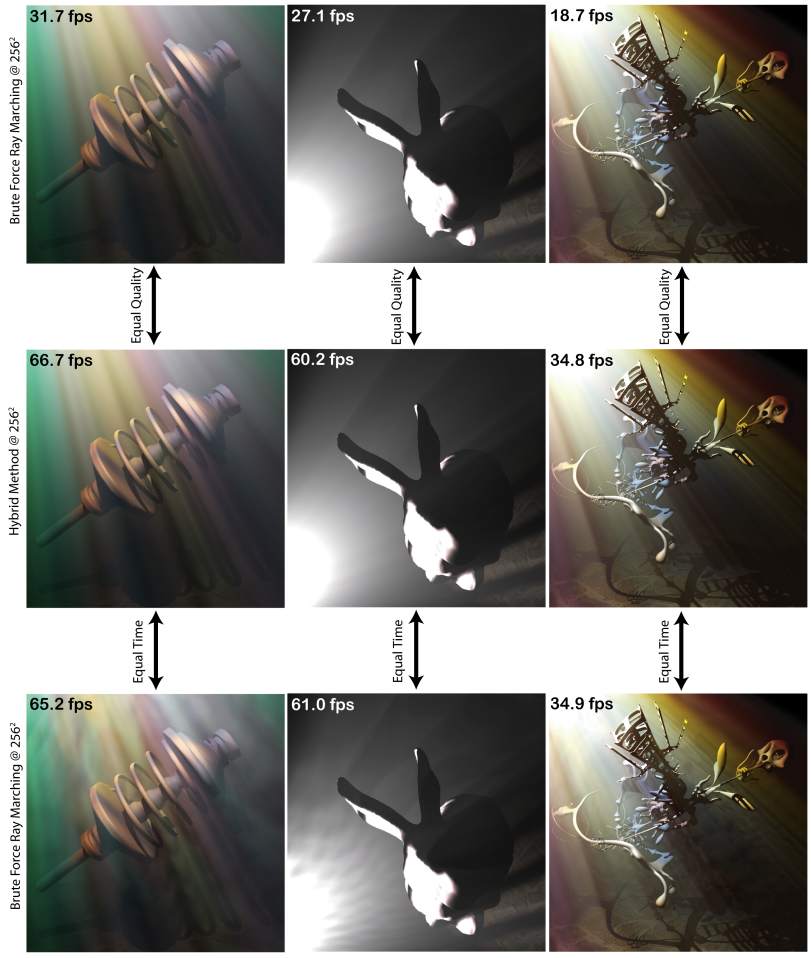

Figure 5: *Table 1 suggests brute force may be equivalent to our hybrid, given identical sampling rates. However, since our hybrid focuses samples in more important regions and trades some blurriness for reduced sampling in large illuminated regions, it gives higher quality for equivalent sampling. Combined with our advantage in per-sample costs seen in Table 1, this gives a significant speed improvement when comparing results of equivalent quality.*

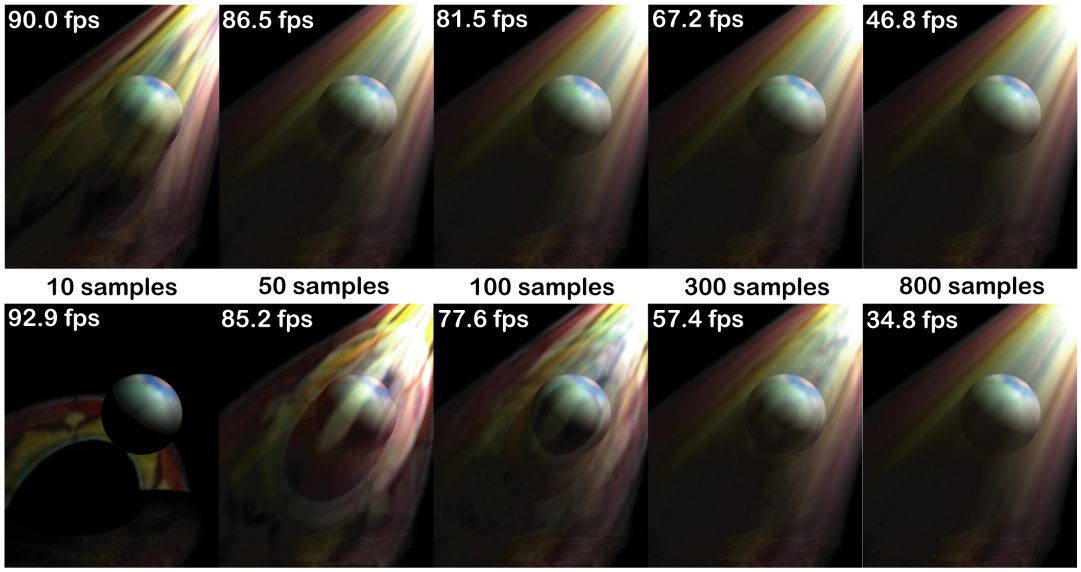

Figure 6: *While Table 1 shows acceptable sampling rates for brute force ray marching with simple point lights, higher sampling rates are required under complex lighting. Even for simple environments such as this illuminated sphere, more than 300 samples are needed, whereas our hybrid performs well with 75 to 100 samples.*

Scene & # Triangles (results in frames per second)	Timings for final images rendered at 1024^2						
	Single colored light source					Textured light	
	No Volumetric Shadows	Brute force ray marching, 150 spp		Ray marching in shadow volume, 50 spp		Ray marching in spotlight volume, 150 spp	
		@1024^2	@256^2	@1024^2	@256^2	@1024^2	@256^2
Sphere (20k)	120.5	12.9	59.8	85.0	99.6	45.5	83.5
Elephant (24k)	137.2	12.6	57.7	89.6	101.1	33.5	78.2
Spring (32k)	160.7	14.6	73.1	76.8	112.5	18.0	72.5
Bunny (70k)	135.0	11.9	61.6	47.5	87.5	32.3	81.3
Fairy (155k)	121.9	12.3	63.5	38.5	72.1	35.9	78.3
Buddha (250k)	96.1	12.2	46.9	50.2	61.4	28.3	61.7
YeahRight (755k)	57.3	11.1	32.8	26.3	33.8	16.1	40.9

Table 1: *Comparisons of our hybrid rendering speed with brute force ray marching and a foggy rendering without volumetric shadows. All framerates are for* 1024^2 *output images, with scattering sampled at either full* 1024^2 *or low* 256^2 *resolution. To allow meaningful comparisons, all scenes were sampled at the same rate for this table. This sampling frequency was chosen so that most scenes were rendered roughly without sampling artifacts, oversampling simple scenes (e.g., the sphere) and undersampling complex scenes (e.g., YeahRight) by about a factor of 2.*

using "light volumes" instead of shadow volumes to identify regions of interest for scattering. In scenes with even more complex shadowing, such as shadows from chain link fence or tree leaves, this technique essentially degenerates to ray marching throughout the volume (with additional shadow volume costs). This situation is similar to that shown by the "Spring" and "YeahRight" scenes where nearly the entire length of each ray is illuminated, though complex fences and leaves would also increase shadow volume costs. One solution would treat complex shadows as a texture, applying Section 3.2 and only using a subset of the shadow volumes in regions where quality shadows are vital.

An alternative would limit objects casting volumetric shadows to geometry near the viewer. Our experience shows volumetric shadows are relatively subtle; in fact, our results use high scattering coefficients and light intensities simply to give clearly visible shadows in static images. In such foggy environments, even objects a few meters away are invisible, so their shadows certainly will not be discernible. In less dense media the shadows from even nearby geometry are quite subtle, so distant shadows may be extraneous.

6 Conclusions and Future Work

This paper presented a prototype hybrid combining ray marching and shadow volume approaches for interactively rendering volumetric shadowing in single-scattering participating media models. We extend this approach to textured light sources and provide a number of optimizations that allow for dramatic speed improvements for only a modest drop in quality. Our approach runs in real-time for simple models and scenes and remains interactive for highly complex objects and scenes with textured spotlights. As our technique is based upon ray marching, rendering quality is comparable to ray tracing using identical scattering models and could, in fact, be used to help accelerate traditional ray tracers. However, since we utilize cheap single-scattering models our results are not comparable to offline images rendered with accurate multi-scattering transport models.

A number of future directions exist, including using a shadow map to directly extrude shadow volumes [14]. This would eliminate all ray marching inside shadowed regions, dramatically improving performance. However initial experimentation has shown this may be tricky, since any errors in extruding this volume result in clearly visible artifacts. Another direction could examine the use of culling techniques [13] to further reduce ray marching; some culling techniques appear directly applicable to volumetric shadows, but many assume shadows only affect surfaces. Finally, further work would be necessary for scenes specular geometry or incorporating media with multiple scattering models.

References

[1] J. Arvo and T. Aila. Optimized shadow mapping using the stencil buffer. *Journal of Graphics Tools*, 8(3):23–32, 2003.

[2] U. Assarsson and T. Akenine-Möller. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics*, 22(3):511–520, 2003.

[3] V. Biri, D. Arques, and S. Michelin. Real time rendering of atmospheric scattering and volumetric shadows. *Journal of WSCG*, 14:65–72, 2006.

[4] E. Chan and F. Durand. An efficient hybrid shadow rendering algorithm. In *Eurographics Symposium on Rendering*, pages 185–196, 2004.

[5] Y. Dobashi, T. Yamamoto, and T. Nishita. Interactive rendering of atmospheric scattering effects using graphics hardware. In *Graphics Hardware*, pages 99–107, 2002.

[6] W. Donnelly and A. Lauritzen. Variance shadow maps. In *Symposium on Interactive 3D Graphics and Games*, pages 161–165, 2006.

[7] A. S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers, Inc., 1994.

[8] N. Hoffman and A. Preetham. *Game Programming Methods*, chapter Real-time light-atmosphere interactions for outdoor scenes, pages 337–352. Charles River Media, 2003.

[9] T. Imagire, H. Johan, N. Tamura, and T. Nishita. Anti-aliased and real-time rendering of scenes with light scattering effects. *The Visual Computer*, 23(9):935–944, 2007.

[10] R. James. *Graphics Programming Methods*, chapter True volumetric shadows, pages 353–366. Charles River Media, 2003.

[11] J. Kniss, G. Kindlmann, and C. Hansen. Multi-dimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[12] S. Lefebvre and S. Guy. Volumetric lighting and shadowing. http://lefebvre.sylvain.free.fr/cgshaders/vshd, 2002.

[13] D. B. Lloyd, J. Wendt, N. Govindaraju, and D. Manocha. CC shadow volumes. In *Eurographics Symposium on Rendering*, pages 197–206, 2004.

[14] M. McCool. Shadow volume reconstruction from depth maps. *ACM Transactions on Graphics*, 19(1):1–26, 2000.

[15] R. Mech. Hardware-accelerated real-time rendering of gaseous phenomena. *Journal of Graphics Tools*, 6(3):1–16, 2001.

[16] K. Mitchell. *GPU Gems 3*, chapter Volumetric Light Scattering as a Post-Process, pages 275–285. Addison-Wesley, 2007.

[17] T. Nishita, Y. Miyawaki, and E. Nakamae. A shading model for atmospheric scattering considering luminous distribution of light sources. In *Proceedings of SIGGRAPH*, pages 303–310, 1987.

[18] P.-P. Sloan, N. Govindaraju, D. Nowrouzezahrai, and J. Snyder. Image-based proxy accumulation for real-time soft global illumination. In *Proceedings of Pacific Graphics*, pages 97–105, 2007.

[19] B. Sun, R. Ramamoorthi, S. Narasimhan, and S. Nayar. A practical analytic single scattering model for real time rendering. *ACM Transactions on Graphics*, 24(3):1040–1049, 2005.

[20] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *IEEE International Conference on Computer Vision*, pages 839–846, 1998.