Interactive Volume Caustics in Single-Scattering Media

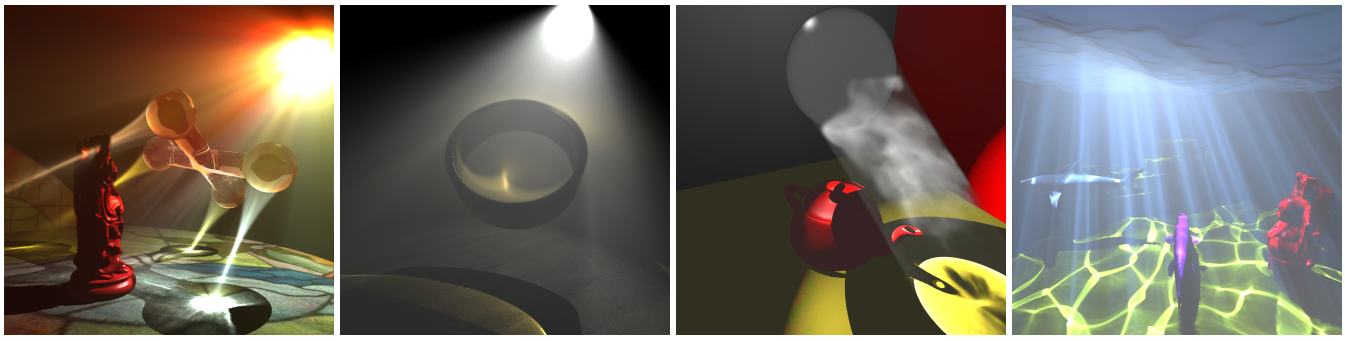
Wei Hu[1] Zhao Dong[2] * Ivo Ihrke[3] Thorsten Grosch[4] Guodong Yuan[1] Hans-Peter Seidel[2]

[1]Beijing University of Chemical Technology China

[2]MPI Informatik Germany

[3]University of British Columbia Canada

[4]Universität Magdeburg Germany



(a) 31 fps (b) 28 fps (c) 11 fps (d) 12.5 fps

Figure 1: *Different rendering results generated by our screen-based interactive volume caustics method. Both, specular and refractive volume caustics in homogeneous and inhomogeneous participating media are handled by our technique.*

Abstract

Volume caustics are intricate illumination patterns formed by light first interacting with a specular surface and subsequently being scattered inside a participating medium. Although this phenomenon can be simulated by existing techniques, image synthesis is usually non-trivial and time-consuming.

Motivated by interactive applications, we propose a novel volume caustics rendering method for single-scattering participating media. Our method is based on the observation that line rendering of illumination rays into the screen buffer establishes a direct light path between the viewer and the light source. This connection is introduced via a single scattering event for every pixel affected by the line primitive. Since the GPU is a parallel processor, the radiance contributions of these light paths to each of the pixels can be computed and accumulated independently. The implementation of our method is straightforward and we show that it can be seamlessly integrated with existing methods for rendering participating media.

We achieve high-quality results at real-time frame rates for large and dynamic scenes containing homogeneous participating media. For inhomogeneous media, our method achieves interactive performance that is close to real-time. Our method is based on a simplified physical model and can thus be used for generating physically plausible previews of expensive lighting simulations quickly.

Keywords: volume caustics, ray marching, real-time rendering

*The first two authors contributed equally.

I3D 2010, Washington, DC, February 19 – 21, 2010.
© 2010 ACM 978-1-60558-938-1/10/0002 $10.00

1 Introduction

Global illumination effects increase the realism of computer generated scenes significantly. Caustics caused by specular or refractive objects are stunning visual effects, even more so in participating media, where *volumetric caustics* can be observed, see Fig. 1.

However, most existing methods for computing volumetric caustics are computationally expensive, preventing interactive applications from including this appealing effect. In this paper we propose a novel interactive volume caustics rendering method for single-scattering participating media. We derive a simplified physics-based model enabling the efficient rendering of volumetric caustics in participating media exhibiting variations in scattering and absorption coefficients as well as the, potentially anisotropic, scattering phase function. We describe a practical GPU-based implementation and evaluate the technique in detail.

Our method avoids all pre-computations, enabling the interactive simulation of light interaction with fully dynamic refractive and reflective objects while maintaining good temporal coherence in animated renderings. Since large and complex scenes can be handled efficiently by our technique, the rendering of volumetric caustics in interactive applications like computer games is becoming an option. Additionally, our method is applicable for fast preview generation of a more complex lighting simulation like volumetric photon mapping [Jensen and Christensen 1998], as required e.g. for feature films and in commercial rendering packages.

In brief, we present the following contributions:

1. We derive a theoretically grounded simplified image formation model for volume caustics in single-scattering media, and,
2. Based on a reformulation of the proposed model, we develop a screen-based interactive rendering technique that uses line primitives [Krüger et al. 2006; Sun et al. 2008] to efficiently splat radiance contributions to image pixels.

The remainder of the paper is organized as follows: First, we re-

view previous work in Section 2. Section 3 gives a short overview of our method. We then derive a simplified image formation model for volume caustics in single-scattering media, Section 4, the implementation of which on graphics hardware is covered in Section 5. Section 6 presents experimental results and comparison against existing techniques. We then conclude the paper in Section 7 and discuss avenues for future research.

2 Related Work

Caustics are phenomena caused by the focusing and de-focusing of light rays upon interaction with specular surfaces. Both, specularly reflective and specularly refractive objects give rise to complex volumetric light distributions. The interaction of these light distributions with opaque surfaces results in *surface caustics* while the effect of their interaction with *participating media* is known as *volume caustics*. The rendering literature on the subject can be grouped accordingly.

Surface Caustics are generated by rays that refocus on diffuse surfaces after refraction or reflection on a specular surface. In computer graphics, photon mapping [Jensen 2001] is the gold standard technique to successfully simulate this kind of phenomenon. Relying on the fast processing speed of modern GPUs, several methods were developed to approximate reflection [Estalella et al. 2006; Umenhoffer et al. 2007; Yu et al. 2005] and refraction [Oliveira and Brauwers 2007; Davis and Wyman 2007; Wyman 2005] effects in real-time. While the aforementioned techniques were developed for rendering reflected and refracted viewing rays, they are equally well applicable for fast photon path calculations on graphics hardware.

The first GPU photon map implementation was described by Purcell et al. [2003]. Recently, several methods for approximating the effect of surface caustics on the GPU have been proposed [Shah et al. 2007; Hu and Qin 2007; Wyman and Davis 2006; Szirmay-Kalos et al. 2005]. As in normal photon mapping, these methods follow specularly reflected and refracted photons from the light source and store their hit positions into a so called photon buffer. A second pass then re-organizes this information and splats it into a caustic map, which is then projected onto the scene similar to shadow mapping. In [Wyman 2008; Wyman and Nichols 2009], the speed and quality of this basic technique is improved using a hierarchical data structure to discard non-contributing and redundant photons. Yu et al. [Yu et al. 2007] present a real-time caustics rendering method based on image space computations, modeling the effect of specular objects as a distorted generalized linear camera model.

Participating Media introduce global shading effects for viewing rays. Instead of shading single surface points based on an approximation of the incident radiance, line integrals over viewing rays have to be computed. Computing the incident radiance is significantly complicated by the effect of multiple scattering [Kajiya and Von Herzen 1984].

Volumetric photon mapping [Jensen and Christensen 1998] decouples radiance transport from the light source into the medium and the integration step to determine the radiance of the viewing ray. The final gathering step along a ray can be computed directly per ray [Jarosz et al. 2008] or in screen space [Boudet et al. 2005]. A complete description of off-line methods for rendering participating media is, however, beyond the scope of this paper; for a good overview we refer the interested reader to [Cerezo et al. 2005].

Typical applications of real-time rendering in the presence of participating media are the visualization of clouds [Dobashi et al. 2000; Harris and Lastra 2001] and smoke [Ren et al. 2008; Zhou et al. 2008]. Approximating the shafts of light, that are a typical effect of single scattering, can be achieved by blending layered materials [Dobashi et al. 2002] or by warping volumes [Iwasaki et al. 2002]. Sun et al. propose an analytical airlight model for real-time single-scattering in homogeneously scattering media [Sun et al. 2005]. This work was extended to real-time rendering of volumetric shadow regions by Wyman and Ramsey [2008]. Our work uses this technique to render the shadow regions surrounding volumetric caustics.

Volume Caustics can be computed using the volumetric photon mapping technique [Jensen and Christensen 1998], an extension to standard photon mapping [Jensen 2001], by tracing and storing photons throughout the volume covering the participating medium. Volume caustics are typically seen in under water scenarios where shafts of light are visible due to focusing of light rays by the water surface. Rendering of these effects was demonstrated by Nishita et al. [1994] and by Ernst et al. [2005] using a triangle-based caustics volume reconstruction method that is very geometry-intensive.

The eikonal rendering technique proposed by Ihrke et al. [2007] uses a volumetric object description to render most of the effects of refraction and participating media in real time. However, if the lighting conditions or scene geometry change, several seconds are required for the re-computation of the incident radiance distribution. Sun et al. [2008] modify the previous technique by combining it with a photon mapping algorithm. They achieve fully dynamic rendering of refraction, absorption and single-scattering effects in participating media at interactive frame rates. Volumetric object representations, however, induce a discrete representation of refractive objects and tend to blur surface and caustic details.

Lines as Rendering Primitives are often used in visualization of vector field data [Zöckler et al. 1996; Mallo et al. 2005]. They are rarely employed as general primitives, even though their use has been advocated by some authors [Wong et al. 2005] and efficient applications in natural scene rendering have been described [Deussen et al. 2002].

Recently, the interactive global illumination literature has produced some applications of lines as fast intermediate rendering primitives. Krüger et al. [2006] demonstrate a screen-based surface and volume caustic rendering technique. The authors directly splat energy to screen pixels using refracted lines as querying primitives. The focus is on surface caustic rendering even though some results are shown for volume caustics as well. Another, connected technique by Sun et al. [2008] also uses lines between specular object and receiver as rendering primitives. Here, however, the lines are rasterized into an intermediate illumination volume, similar to [Ihrke et al. 2007], which enables a correct evaluation using a second ray marching step.

Our work can be seen as a combination of the previous two techniques. We combine the strength of the screen-based approach, which is very high resolution, high performance rendering at low memory foot-prints, with the physical accuracy of the photon-based approach, which in its original implementations is either too slow for real-time applications [Jensen and Christensen 1998] or yields blurry results [Ihrke et al. 2007; Sun et al. 2008] due to high memory consumption and thus limited resolution.

Contemporary to our work, [Papadopoulos and Papaioannou 2009] presented real-time caustics and godrays for underwater scenes using an implementation which is similar to the method of Krüger et al. [2006]. In contrast to our work, no comparison to ground truth is shown and only a homogeneous medium is supported.

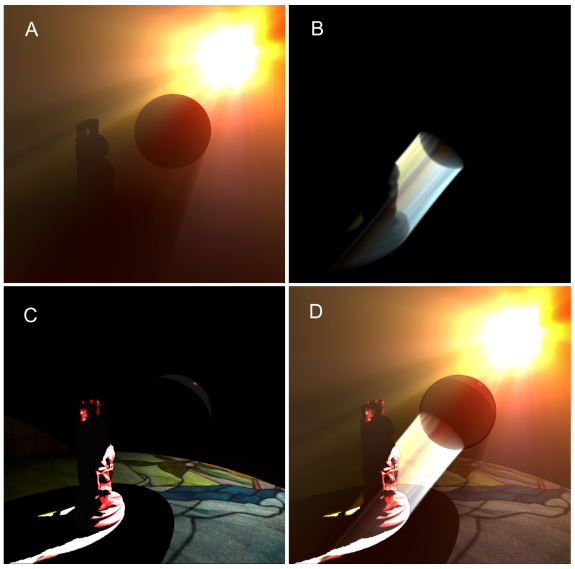


Figure 2: *The final image (D) is composed of an airlight image with a shadow volume (A), a volume caustic image (B) and the illumination of the surface (C).*

3 Overview

Image formation for volumetric caustics is an involved process. We thus start with an overview of our algorithm, Fig. 2. The radiance estimate for an image pixel in the presence of a participating medium and specular objects can be split into three separate components:

1. radiance scattered into the viewing direction by the participating medium for incident rays *not* having interacted with specular objects (A),

2. radiance scattered into the viewing direction from incident rays that experienced specular reflection or refraction events prior to the scattering event (B), and

3. surface radiance, possibly illuminated by a caustic (C).

Since superposition of light is linear, the final image (D) can be computed by summing the three components. For steps (A) and (C) we employ previously developed algorithms. Our focus in this paper is on the efficient computation of component (B).

On a coarse level, our technique employs the following steps: First, we render the airlight and volumetric shadow contributions (A) using the anisotropic version of Sun et al.'s model [2005] computed with the algorithm of Wyman and Ramsey [2008]. Second, the image with the volume caustics (B) is generated by computing the light paths from the light source that are reflected or refracted at a specular object. We draw these paths as *lines* and directly compute radiance contributions for each of the affected pixels which are summed up over all line segments representing light rays. The algorithm can be seen as a radiance splatting operation that emulates GPU *ray marching*. This rendering pass is our main contribution and it is described in detail in the following sections. Third, we generate an image of the surface illumination and the surface caustics (C) using Wyman's hierarchical caustic map (HCM) algorithm [Wyman 2008]. The final image is then the sum of these three images.

L_{in}	incoming radiance in caustic volume
L_{ls}	light source radiance
$\sigma_s(\mathbf{x})$	scattering cross-section
$\sigma_a(\mathbf{x})$	absorption cross-section
$\kappa(\mathbf{x})$	extinction coefficient $\kappa = \sigma_a + \sigma_s$
$\Omega_0(\mathbf{x})$	albedo of participating medium $\Omega_0(\mathbf{x}) = \frac{\sigma_s}{\kappa}$
$\tau(\mathbf{a},\mathbf{b})$	transmittance from $\mathbf{a}$ to $\mathbf{b}$: $\exp\left(-\int_{\mathbf{a}}^{\mathbf{b}} \kappa(\mathbf{x})dx\right)$
p	scattering phase function
$d_{\mathbf{ab}}$	distance between points $\mathbf{a}$ and $\mathbf{b}$
T	Fresnel transmittance (or reflectance)

Figure 3: *A summary of the notation used in this paper.*

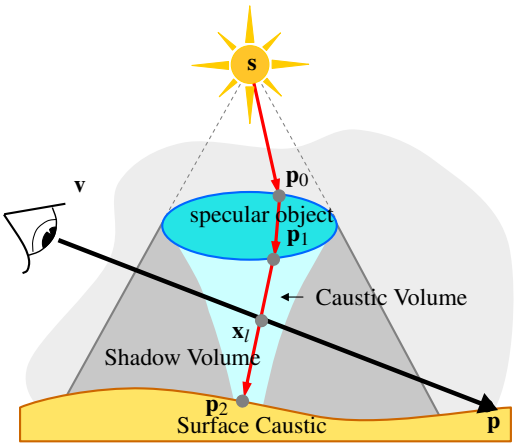


Figure 4: *A volume caustic can be generated by computing all rays intersecting a specular object. The caustic can be rendered by drawing lines from exit points* $\mathbf{p}_1$ *to surface points* $\mathbf{p}_2$ *for all light paths interacting with the object.*

4 Line-Based Volume Caustics

Our goal in this section is to derive a physically accurate model for volume caustics in single scattering media. We concentrate on the radiance contributed to the image by single scattering in the caustic volume, i.e. step (B) in Fig. 2.

In the presence of a participating medium the ray integral for a viewer at position $\mathbf{v}$ looking into direction ω is given by

$$L(\mathbf{v},\omega) = \int_{Ray} \tau(\mathbf{v},\mathbf{x})\sigma_s(\mathbf{x}) \int_{\Omega} p(\mathbf{x},\omega,\omega')L_{in}(\mathbf{x},\omega')d\omega' dx, \quad (1)$$

where τ denotes transmittance, σ_s the scattering cross section, and p the scattering phase function (see also Fig. 4). Eq. 1 requires the computation of the incident radiance distribution L_{in} at every point $\mathbf{x}$ along a viewing ray.

Since radiance, in the absence of ray attenuation, is an optical invariant along the ray even when passing refractive objects [Veach 1998], L_{in} can be determined efficiently by texture look-ups for a light source texture or by simply setting radiance values. Eq. 1 can be discretized in a straight-forward manner as

$$L(\mathbf{v},\omega) \approx \sum_i \tau(\mathbf{v},\mathbf{x}_i)\sigma_s(\mathbf{x}_i) \sum_j p(\mathbf{x}_i,\omega,\omega'_j)L_{in}(\mathbf{x}_i,\omega'_j)\Delta\omega\Delta x, \quad (2)$$

where Δx is the constant step size of the ray marching integral and $\Delta\omega$ is the size of the discretized solid angle. Radiance is summed over spatial positions along the viewing ray $\mathbf{x}_i$ in all directions ω'_j.

Since we are rendering line segments to connect light paths between viewing rays and light source, see Fig. 4, the double sum in Eq. 2 can be converted into a single sum over these line segments:

$$L(\mathbf{v},\omega) \approx \sum_l \tau(\mathbf{v},\mathbf{x}_l)\sigma_s(\mathbf{x}_l)p(\mathbf{x}_l,\omega,\omega'_l)L_{in}(\mathbf{x}_l,\omega'_l)\Delta\omega\Delta x_l. \quad (3)$$

Each line segment intersecting the viewing ray introduces a light path between $\mathbf{s}$ and $\mathbf{v}$. The connection is made at point $\mathbf{x}_l$ where the line segment passing in direction $\omega'_l = \mathbf{p}_2 - \mathbf{p}_1$ intersects the viewing ray in direction ω. The radiance reaching point $\mathbf{x}_l$ from $\mathbf{s}$ is equal to the original radiance at the light source attenuated by absorption and out-scatter along the ray. Thus we have an additional factor of $\tau(\mathbf{s},\mathbf{x}_l)$ that includes the Fresnel factors T for specular reflection or refraction, e.g. $L_{in} = L_{ls}\tau(\mathbf{s},\mathbf{p}_0)T_0\tau(\mathbf{p}_0,\mathbf{p}_1)T_1\tau(\mathbf{p}_1,\mathbf{x}_l)$ in the case of two-bounce refraction. Note that the conversion of Eq. 2 into Eq. 3 introduces the implicit assumption that the radiance distribution L_{in} in the caustic volume is zero for light rays not being represented by line segments, i.e. all light leaving the light source at $\mathbf{s}$ is reaching $\mathbf{x}_l$ via a light ray that can be represented by a line segment.

We assume the discretized solid angle $\Delta\omega$ to be constant. This is the case if the area of the light source is small with respect to the distance $d_{\mathbf{sx}_l}$ since the size of the solid angle under which the light source appears at $\mathbf{x}_l$ is approximately constant for comparatively minor path length differences between $\mathbf{s}$ and $\mathbf{x}_l$ for different incident directions.

Note that using lines as rendering primitives, we cannot ensure an even sampling of the radiance function L_{in} along the viewing ray. We cannot predict the intersection points $\mathbf{x}_l$ without rendering all line segments and storing the intersection points with all viewing rays. This would require an intermediate storage of the irradiance as in [Ihrke et al. 2007; Sun et al. 2008]. Thus, we cannot compute an appropriate variable spatial step size Δx_l, Fig. 5 (left).

Therefore, in our algorithm, we simulate a constant step size $\Delta x_l = \text{const.} = \Delta x$ for the ray marching integral, where Δx is a user parameter emulating a fixed step size as found in explicit ray marching algorithms. We re-sample the radiance function L_{in} on-the-fly while rendering the line segments. Using a filtering operation, we re-distribute the radiance function to the closest regular step points, see Fig. 5 (right). We employ a normalized Gaussian filter [Jensen 2001], the support of which can be modified as a user parameter r. We thus have derived a simplified image formation model based on physical assumptions. Our model intrinsically supports anisotropic scattering phase functions as well as inhomogeneously scattering participating media via spatially varying scattering phase functions, absorption, and scattering coefficents. Note that the case of homogeneous media is typically more efficient in terms of implementation since the evaluation of $\tau(\mathbf{x}_i,\mathbf{v})$ does require the sampling of a line integral in the case of inhomogeneous media.

To summarize, our simplifying assumptions are

- The image formation is dominated by the effect of single scattering, and
- The distance between light source and volume caustic points is large compared to the size of the light source.

In the following section we describe how this simplified model can be efficiently implemented on the GPU.

5 Implementation

In this section we discuss the implementation of our simplified image formation model. During the discussion we refer to Fig. 6 which shows an outline of our algorithm. We will refer to illuminating rays, such as $\mathbf{p}_1$,$\mathbf{p}_2$ in Fig. 4, as *light rays* whereas rays from the camera are called *viewing rays*. Our algorithm proceeds in two main stages. The first stage is the computation of three-dimensional light ray segments. The second stage then draws all those segments to the screen buffer and blends the radiance contributions according to Eq. 3 while emulating a proper ray marching implementation via filtered re-distribution of radiance values to neighboring regular step points, Fig. 5.

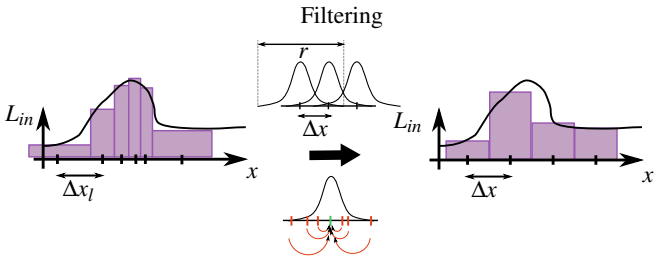


Figure 5: *We simulate regular interval spatial integration along the viewing ray. For this, the irregular radiance samples (with a variable step size of Δx_l) caused by line rendering (left diagram) are converted via filtering into a regular representation with fixed step size Δx (right). Note that this fixed step size representation is implicit. In practice, we directly accumulate the filtered values to compute the pixel integral. The small figure above the arrow shows that the Gaussian filters with support r are centered at the regular sample points to be emulated. The lower figure indicates how the irregular samples obtained from line rendering (red) contribute to a simulated regular sample point (green). All irregular samples within the support of the filter function contribute to the corresponding regular sample point.*

5.1 Generating Line Primitives

In this stage we compute segments of light rays after their interaction with a specular object and before hitting a diffuse surface. These line segments, if passing through a participating medium, will generate an indirect light path between light and viewing rays by means of a scattering interaction.

In practice, we simulate one- or two-bounce reflection or refraction, respectively. For this purpose we generate a depth map of the scene excluding specular objects as seen by the light source (step 1, bottom). Additionally, we generate front- and back-facing depth and normal maps of the specular objects (step 1, upper rows). The specular object depth and normal buffers are then used to compute the light rays that are reflected or refracted from the object [Hu and Qin 2007]. This step results in a position $\mathbf{p}_1$ on the back-face of the object as well as a light ray direction after specular interaction. An additional intersection of these rays with the scene depth map results in $\mathbf{p}_2$. If no intersection is found, we intersect the ray with a large bounding sphere surrounding the scene. In this way, missing geometry in the depth map does not invalidate the computed light rays.

This procedure results in the two points $\mathbf{p}_1$ and $\mathbf{p}_2$, Fig. 4, and thus determines the light ray segment in camera space. Fig. 6, step 2, visualizes this *volume caustic buffer*. The image below the buffers shows a 3D rendering of the end-point positions as seen from the camera. In our implementation we store $\mathbf{p}_1$ and $\mathbf{p}_2$ into the same texel position of two different textures. In a subsequent pass we render point primitives to achieve a read back of $\mathbf{p}_1$ and $\mathbf{p}_2$ from the textures into the geometry shader stage. This way, a line primitive can be constructed. Initially, the coordinate values of the end

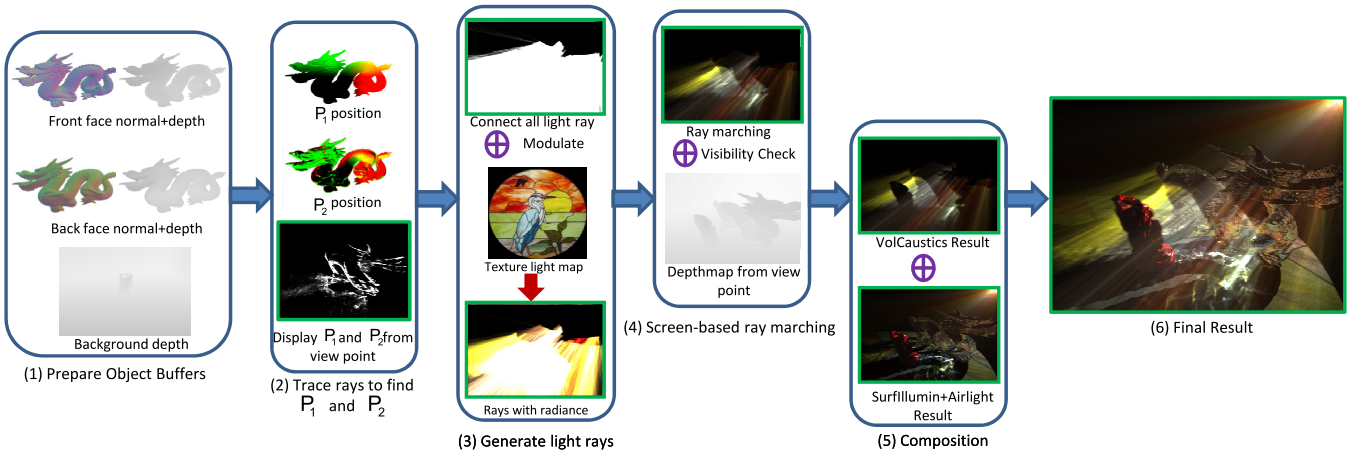


Figure 6: *Algorithmic steps of our algorithm.*

points are in light space; we transform them into camera space in this stage. After the geometry shader, the newly created line primitive will be automatically rasterized before entering the pixel shader stage.

5.2 Light Ray Blending

In step 3, we modulate all light rays with a light source texture. Note that a rendering of all lines with a radiance value picked directly from the texture would result in over-exposed images as shown in the image at the bottom of step 3.

To properly compute the correct radiance contributions we employ a fragment shader implementing Eq. 3. This step requires the use of the front- and back-facing object buffers from step 1, the volume caustic buffer, step 2, and the modulation texture from step 3. They all share the same resolution and corresponding pixels in the buffers describe different properties of the light ray $\mathbf{s}, \mathbf{p}_2$. The fragment shader first computes the point of intersection $\mathbf{x}_l$ between the viewing ray and the light ray. Next, to properly emulate the regularly discretized ray marching integral, Fig. 5 , we compute the regular step points along the ray that are within the filter support.

This results in a set $\{\mathbf{x}_i | i = 1 \ldots N\}$ of regularly spaced points in the vicinity of $\mathbf{x}_l$. N is typically in the range of $2-4$. For each point in this set we determine the light path described by the vertices $(\mathbf{v}, \mathbf{x}_i, \mathbf{p}_1, \mathbf{p}_0, \mathbf{s})$ and compute per segment attenuation and Fresnel transmission factors. Combining the attenuation factors and the light source radiance from the modulation texture, step 3, weighted by the kernel value, we obtain the incident radiance value L_{in} at $\mathbf{x}_i$. The ray direction remains $\omega'_l = \mathbf{p}_2 - \mathbf{p}_1$. This way, we can directly evaluate parts of the sum in Eq. 3 and add the appropriate radiance contribution to the pixel value.

5.3 Visibility and Remaining Illumination Components

In step 4 we compute visibility of the light rays as seen from the camera. This step requires a scene depth map in camera coordinates. The depth map includes the specular objects. We simply cull light ray fragments if they are behind objects. This step obviously introduces artifacts; refractive objects appear opaque and do not transmit light from volume caustics. This limitation is inherent in our screen-based approach. Since we do not store the irradiance distribution in the caustic volume we cannot perform volume rendering along refracted viewing rays.

Finally, we add the images containing the airlight and volumetric shadows, Fig. 2 (A), and the surface illumination, Fig. 2 (C). Note that in order to compute the radiance contribution due to the surface properly, the transmittances $\tau(\mathbf{s}, \mathbf{p})$ and $\tau(\mathbf{v}, \mathbf{p})$ have to be computed and multiplied to the surface radiance in absence of a participating medium.

5.4 Inhomogeneous Media

The previous description applies to the case of homogeneous media. The case of inhomogeneous media differs only slightly. Instead of computing transmittance values $\tau(\mathbf{a}, \mathbf{b})$ analytically, we now have to perform sampling and numerical integration in order to retrieve the inhomogeneous information. It might seem that this excludes the simulation of inhomogeneous media at interactive frame rates.

Note, however, that sampling is only required to compute the transmittance. The accumulation of varying albedo and phase function is not affected by the quality of the numerical integration since it is computed implicitly by summing line segment contributions. Thus, in practice, we can choose a very low sampling rate for the numerical integration. Although this assumes no high-frequency changes inside the medium, the poor approximation will only be visible in the shadow cast by the inhomogeneous medium and in slight low-frequency intensity variations within the caustic. The latter effect is usually masked by the inhomogeneous appearance and the brightness of the caustic regions, see Fig 11.

6 Results and Discussion

In this section we report on the quality and performance of our method. Our technique was implemented in OpenGL and all results were rendered on a Intel Core2 Quad Processor Q9550 with 2.83GHz using an NVIDIA GeForce 280 GTX graphics card. Except where explicitly noted, all the rendering results in the paper are generated using a volume caustics buffer resolution of 1024×1024 pixels. Similarly, the size of the screen buffer is usually 1024×1024, except in the test exploring the performance impact of the screen buffer size. We use a volume shadow buffer resolution [Wyman and Nichols 2009] of 256×256 pixels; the size of the surface caustics buffer [Wyman 2008] is 1024×1024.

6.1 Ground Truth Comparison

To verify the accuracy of our method, we compare volume caustics rendered with our model to both ground truth images generated with Mental Ray and images generated using the previous techniques by [Krüger et al. 2006] and [Papadopoulos and Papaioannou 2009]. Since the latter two techniques are very similar, and [Krüger et al. 2006] does not contain explicit formulas we implement attenuation along the ray as in [Papadopoulos and Papaioannou 2009] but refer to the technique itself as [Krüger et al. 2006]. The results of this comparison are shown in Fig. 7. Since [Krüger et al. 2006] does not support anisotropic and inhomogeneous media, the test scene is isotropic and homogeneous.

As can be seen from the figure, our results closely match the images rendered with photon mapping (5 min. computation time). Our technique , in comparison, runs at more than 25 fps. For the simple case of isotropic and homogeneous scenes, the technique of Krüger et al. [2006] is also able to generate a similar image with slight advantages in rendering speed. The performance penalty associated with our more general and physically accurate model is about 5% for the volume caustics stage compared to Krüger et al. [2006]. The overall performance drop in the full algorithm is not noticable. Note, that it is necessary to scale the output of [Krüger et al. 2006] linearly to match the ground truth rendering and that the scale factor is scene dependent and cannot be easily estimated. Our method, on the other hand, runs with fixed parameter settings of $\Delta x = 0.2$.

The differences between our results and the ground truth rendering can mainly be attributed to the approximate computation of the light path segments in our algorithm. Note, however, that future improvements of GPU ray tracing directly increase the accuracy of our method since the computation of the light rays is an independent module. A second difference is that photon mapping can determine light contributions seen *through* refractive objects, which is impossible for our algorithm as discussed previously. A third effect is a slight multiple scattering component in the ground truth image whereas our result shows only single scattering according to our model. Finally, differences can appear since we only follow the most important light paths (eg. two refractions inside the object).

A comparison between ground truth and our algorithm for the anisotropic case is shown in Fig. 8. While there are differences in the images, mainly concerning brightness and sharpness of the caustics, the overall characteristic of the medium is captured by our technique. The smoother look of the ground truth result can be attributed to multiple scattering effects.

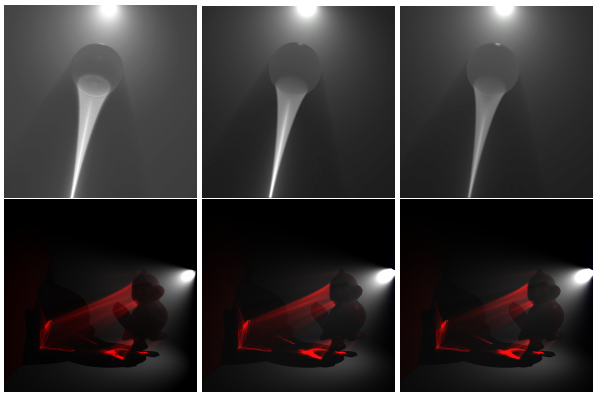


Figure 7: *Comparison of volumetric photon mapping as implemented in MentalRay (left), our method (middle), Krüger et al. [2006] linearly scaled with a manually determined value to match ground truth (right, scale factors are 0.06 and 0.02 for the top and bottom results). The ground truth results are generated with Mental Ray, using 2 million photons and take around 5 minutes to compute. All results generated by our method use a* 1024×1024 *volume caustics buffer and can be rendered with more than 25 fps.*

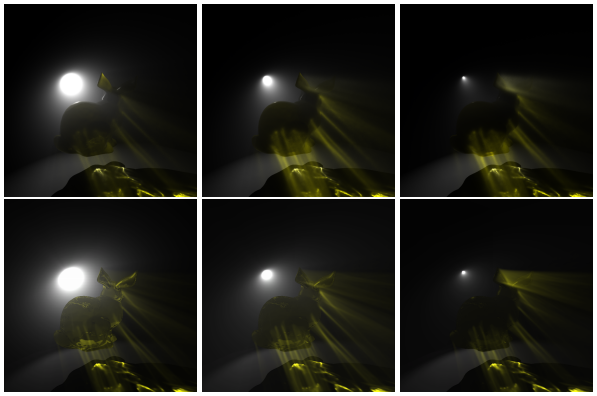


Figure 8: *Comparison of volume photon mapping in anisotropic participating media (top row) with results of our method (bottom row). The phase function uses the Henyey-Greenstein model as implemented by MentalRay. The anisotropy parameters are* $g = 0.7$ *(left),* $g = 0.0$ *(middle), and* $g = -0.7$ *(right) for forward, isotropic and backward scattering, respectively.*

6.2 Performance Analysis

To assess the performance characteristics of our technique we experimented with three test scenes shown in Fig. 9. The scenes are increasingly complex in terms of volume caustic computation. The ring scene demonstrates one-bounce reflection, the Buddha scene is rendered with two-bounce refraction, and the gemstone scene includes both effects. The timing data in Table 1 shows the rendering time for each rendering stage. We can see that only rendering the volumetric caustics runs at over 80 fps on average when using a caustics buffer resolution of 1024×1024. For the gemstone scene, we compute the volume caustics generated by both two-bounce refraction and one-bounce reflection. The total number of caustic buffers doubles in this case. From the timing data in the table, we see that the time required for the surface and volume caustics stages is also approximately doubled compared to the scenes with a single caustic buffer. This suggests that our method scales linearly with the number of light rays involved in the caustics computation. Since HCM [Wyman 2008] contains an additional overhead due to hierarchical processing compared to the volume caustics stage, for usual scene settings with a low number of pixels affected by volume caustics the surface caustics stage dominates rendering time. As a second test, we evaluated the dependence of our algorithm on screen size and caustic buffer resolution. Table 2 shows that an increasing caustics buffer size has more severe consequences than increasing the screen resolution. Of course, this finding depends on

	Rendering Stage				
Scene	AL	SC	VC	Comp.	FPS
Ring	4.5 ms	15 ms	10 ms	4 ms	28
Buddha	5 ms	13 ms	9 ms	4 ms	31
Gemstone	5 ms	28 ms	21 ms	4 ms	17

Table 1: *Timing data of different rendering stages in Fig. 9. AL = airlight and volume shadows , Fig. 2 (A), SC = surface caustics (B), VC = volume caustics (C), Comp. = composition (D), FPS - overall performance.*

	Screen Buffer Resolution		
CB Res.	256^2	512^2	1024^2
256^2	193	173	171
512^2	163	140	115
1024^2	93	74	62

Table 2: *Timing data for different caustic buffer (CB Res.) and screen buffer resolutions. Numbers are for the Buddha scene and are given in frames per second.*

the number of pixels covered by lines. The other two test scenes show similar performance characteristics. Note however, that the timings are for the volume caustics rendering step only. Since this part of our algorithm only consumes about 30% of the overall computation time, the performance gains for lower resolution buffers are less dramatic in a realistic scenario.

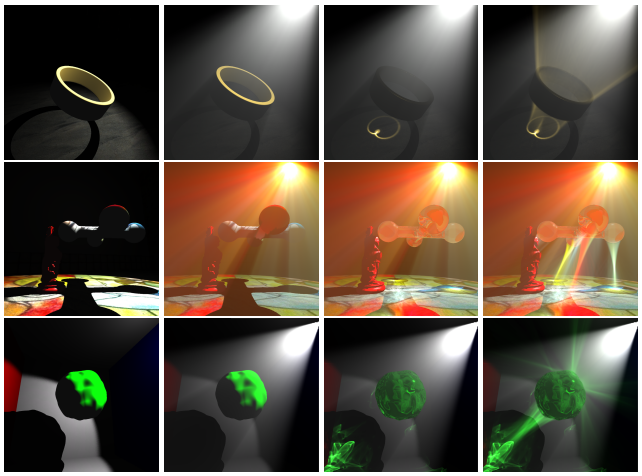
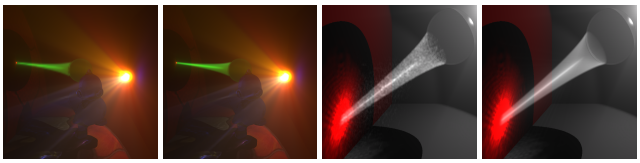


Figure 9: *Different combinations of illumination methods for our test scenes. From left to right, the following type of illumination is included: Direct illumination, airlight and volume shadows, surface caustics, and volume caustics. The ring scene (top row) is rendered with one-bounce reflection, the Buddha scene (middle row) illustrates two-bounce refraction and the gemstone scene (bottom row) includes both one-bounce reflection and two-bounce refraction.*

6.3 Influence of User Parameters

Several parameters affect the image quality and rendering performance of our method. The first is the volume caustics buffer resolution. All results shown in this paper use 1024×1024 as the resolution for the volume caustics buffer. In Fig. 10 (left) we show results with different volume caustics buffer resolutions but keep the surface caustics buffer resolution at 1024×1024 pixels. A higher resolution results in better image quality, however, the volume caustics quality is still reasonable even though details appear slightly blurred in the low resolution case. Another user parameter is the size of the filter kernel r. We typically choose the support of the filter equal to the step size Δx. In Fig. 10 (right) we show results for drastically different settings. If the support of the filter is chosen too low, noise is being generated in the images. Finally, the number of light sources influences the performance of our method. Each light source requires separate caustic buffers. As we have seen in Table 1, the performance drops approximately linearly with the number of caustic buffers. We also observe this behavior in the case of multiple light sources; tests with one to four light sources resulted in the following numbers: 37 fps(1), 21.5 fps(2), 9.25 fps(3), 2.9 fps(4). An additional strength of our method is the temporal coherence: animated and deforming objects can be displayed correctly without flickering. We can deal with arbitrary deformations without any pre-computation. Since images capture this feature of our method inadequately we recommend to watch the accompanying video.



(a) 256^2, 39 fps (b) 1024^2, 29 fps (c) $r = 0.1 \cdot \Delta x$ (d) $r = \Delta x$

Figure 10: *Left: Different volume caustics buffer resolution. Right: Different filter size.*

6.4 Limitations

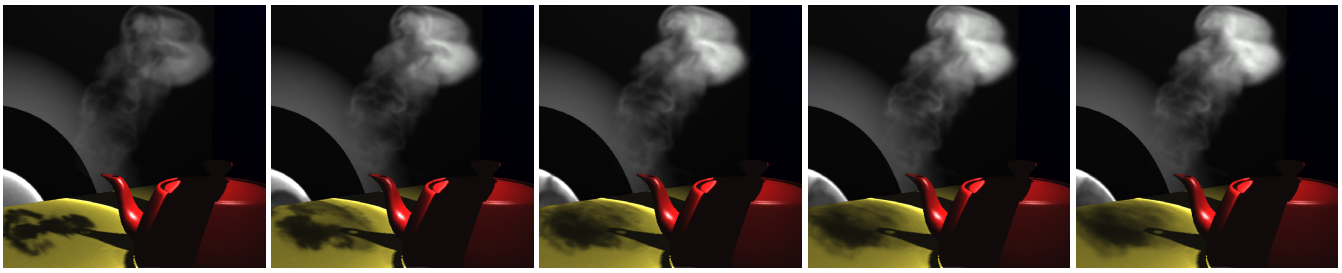
Our screen-based volume caustic technique achieves high performance for highly complex scene settings. However, because of its screen-based nature, the computation cost increases linearly with the effective number of screen pixels that require volume caustics computations. As an example, in the ocean scene, Fig. 1, almost all screen pixels (1024×1024) are involved in volume caustics computations. This is the reason for the low frame rate compared to the other examples.

Another issue is the effective resolution of the caustics buffer. Consider a large field of view for a spot light source illuminating a comparatively small specular object. Since the object occupies very few pixels in the caustic buffer, the rendering quality will suffer in this case. Another, related case is the rendering of extreme close-ups of the volume caustics. Individual lines might become visible in this case. The problem is increased for objects with very high refractive indices since light rays diverge more strongly under these circumstances.

7 Conclusions and Future Work

We have presented a real-time method for rendering volume caustics in single scattering participating media. Our technique generates physically plausible results and enables the rendering of completely dynamic scenes with good temporal coherence. Anisotropic and inhomogeneous media are naturally supported by our algorithm and interactive performance can be achieved in the latter case. Our method simulates the most important light paths for volumetric caustics and introduces a theoretically founded approximation for single scattering media. Since it efficiently renders large and dynamic scenes we believe that it can be applied in computer games and as a preview for more sophisticated lighting simulations.

An interesting direction for future work are hierarchical representations of the caustic buffer. Currently we use separate buffers for surface and volume caustics. However, the two buffers share common information and speed-ups can potentially be gained by combining them into a single buffer. A hierarchical approach similar in spirit to [Wyman 2008] could potentially increase the performance significantly. Another research direction is the approximation of multiple scattering effects in participating media. Since line primitives are performing very well in the case of single scattering media an interesting question is whether this power can be harnessed for multiple scattering approximations.



(a) 2 steps, 41 fps (b) 5 steps, 27 fps (c) 10 steps, 18 fps (d) 20 steps, 10.5 fps (e) 30 steps, 8 fps

Figure 11: *Inhomogeneous smoke rendering using different number of discrete integral steps. Note that a low number of step points only affects the brightness of the volume caustic and the accuracy of the shadow. 10 integration steps already suffice to obtain a visually appealing result with only minor differences to the final solution.*

8 Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. We would like to thank Robert Herzog for very helpful discussions during the development of this paper. We would also like to thank Chris Wyman for making his code publicly available. Thanks to the Stanford University Computer Graphics Laboratory for the Happy Buddha, Dragon and Bunny models. Thanks to Marko Dabrovic for Sibenik model. Thanks to Robert W.Sumner for the horse mesh animations.

Wei Hu is supported by the Young Scientific Researcher Funding of Beijing University of Chemical Technology (No. QN0823), China. Ivo Ihrke is supported by a Feodor-Lynen Fellowship of the Humboldt Foundation, Germany.

References

BOUDET, A., PITOT, P., PRATMARTY, D., AND PAULIN, M. 2005. Photon Splatting for Participating Media. In *Proc. of GRAPHITE*, 375–384.

CEREZO, E., PEREZ-CAZORLA, F., PUEYO, X., SERON, F., AND SILLION, F. 2005. A Survey on Participating Media Rendering Techniques. *The Visual Computer*, 303–328.

DAVIS, S. T., AND WYMAN, C. 2007. Interactive Refractions with Total Internal Reflection. In *Proc. of Graphics Interface*, 185–190.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. 2002. Interactive Visualization of Complex Plant Ecosystems. In *Proc. of IEEE Visualization*, 219–226.

DOBASHI, Y., KANEDA, K., YAMASHITA, H., OKITA, T., AND NISHITA, T. 2000. A Simple, Efficient Method for Realistic Animation of Clouds. In *Proc. of SIGGRAPH*, 19–28.

DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. 2002. Interactive Rendering of Atmospheric Scattering Effects using Graphics Hardware. In *Proc. of Graphics Hardware*, 99–107.

ERNST, M., AKENINE-MÖLLER, T., AND JENSEN, H. W. 2005. Interactive Rendering of Caustics using Interpolated Warped Volumes. In *Proc. of Graphics Interface*, 87–96.

ESTALELLA, P., MARTIN, I., DRETTAKIS, G., AND TOST, D. 2006. A GPU-driven Algorithm for Accurate Interactive Reflections on Curved Objects. In *Proc. of EGSR*, 313–318.

HARRIS, M. J., AND LASTRA, A. 2001. Real-Time Cloud Rendering. *Computer Graphics Forum 20*, 3, 76–84.

HU, W., AND QIN, K. 2007. Interactive Approximate Rendering of Reflections, Refractions, and Caustics. *IEEE Transactions on Visualization and Computer Graphics 13*, 1, 46–57.

IHRKE, I., ZIEGLER, G., TEVS, A., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2007. Eikonal Rendering: Efficient Light Transport in Refractive Objects. *ACM Trans. Graph. 26*, 3, article 59.

IWASAKI, K., DOBASHI, Y., AND NISHITA, T. 2002. Efficient Rendering of Optical Effects within Water using Graphics Hardware. *Computer Graphics Forum 21*, 4, 701–711.

JAROSZ, W., ZWICKER, M., AND JENSEN, H. W. 2008. The Beam Radiance Estimate for Volumetric Photon Mapping. In *Eurographics*, 557–566.

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient Simulation of Light Transport in Scences with Participating Media using Photon Maps. In *Proc. of SIGGRAPH*, ACM, 311–320.

JENSEN, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. AK Peters.

KAJIYA, J. T., AND VON HERZEN, B. P. 1984. Ray Tracing Volume Densities. *Proc. of SIGGRAPH 18*, 3, 165–174.

KRÜGER, J., BÜRGER, K., AND WESTERMANN, R. 2006. Interactive Screen-Space Accurate Photon Tracing on GPUs. In *Proc. of EGSR*, 319–329.

MALLO, O., PEIKERT, R., SIGG, C., AND SADLO, F. 2005. Illuminated Lines Revisited. In *Proc. of IEEE Visualization*, 19–26.

NISHITA, T., AND NAKAMAE, E. 1994. Method of Displaying Optical Effects within Water using Accumulation Buffer. In *Proc. of SIGGRAPH*, vol. 28, 373–379.

OLIVEIRA, M. M., AND BRAUWERS, M. 2007. Real-time refraction through deformable objects. In *Proc. of I3D*, 89–96.

PAPADOPOULOS, C., AND PAPAIOANNOU, G. 2009. Realistic Real-time Underwater Caustics and Godrays. In *Proc. GraphiCon '09*, 89–95.

PURCELL, T., DONNER, C., CAMMARANO, M., JENSEN, H., AND HANRAHAN, P. 2003. Photon Mapping on Programmable Graphics Hardware. In *Proc. of Graphics Hardware*, 41–50.

REN, Z., ZHOU, K., LIN, S., AND GUO, B. 2008. Gradient-based Interpolation and Sampling for Real-time Rendering of Inhomogeneous, Single-scattering Media. *Computer Graphics Forum*, 7, 1945–1953.

SHAH, M. A., KONTTINEN, J., AND PATTANAIK, S. 2007. Caustics Mapping: An Image-Space Technique for Real-Time Caustics. *IEEE Transactions on Visualization and Computer Graphics 13*, 2, 272–280.

SUN, B., RAMAMOORTHI, R., NARASIMHAN, S. G., AND NAYAR, S. K. 2005. A Practical Analytic Single Scattering Model for Real-time Rendering. *ACM Trans. Graph. 24*, 3, 1040–1049.

SUN, X., ZHOU, K., STOLLNITZ, E., SHI, J., AND GUO, B. 2008. Interactive Relighting of Dynamic Refractive Objects. *ACM Trans. Graph. 27*, 3, article 35.

SZIRMAY-KALOS, L., ASZÓDI, B., LAZÁNYI, I., AND PREMECZ, M. 2005. Approximate Ray-Tracing on the GPU with Distance Impostors. *Computer Graphics Forum 24*, 3, 695–704.

UMENHOFFER, T., PATOW, G., AND SZIRMAY-KALOS, L. 2007. *GPU Gems 3*. Addison-Wesley, ch. Robust Multiple Specular Reflections and Refractions, 387–407.

VEACH, E. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Department of Computer Science.

WONG, K.-H., OUYANG, X., LIM, C.-W., TAN, T.-S., AND NIEVERGELT, J. 2005. Rendering Anti-Aliased Line Segments. In *Proc. of CGI*, 198–205.

WYMAN, C., AND DAVIS, S. 2006. Interactive Image-Space Techniques for Approximating Caustics. In *Proc. of I3D*, 153–160.

WYMAN, C., AND NICHOLS, G. 2009. Adaptive Caustic Maps Using Deferred Shading. *Computer Graphics Forum 28*, 2, 309–318.

WYMAN, C., AND RAMSEY, S. 2008. Interactive Volumetric Shadows in Participating Media with Single-Scattering. In *Proc. of IEEE Symposium on Interactive Ray Tracing*, 87–92.

WYMAN, C. 2005. An Approximate Image-Space Approach for Interactive Refraction. *ACM Trans. Graph. 24*, 3, 1050–1053.

WYMAN, C. 2008. Hierarchical Caustic Maps. In *Proc. of I3D*, 163–171.

YU, J., YANG, J., AND MCMILLAN, L. 2005. Real-time Reflection Mapping with Parallax. In *Proc. of I3D*, 133–138.

YU, X., LI, F., AND YU, J. 2007. Image-Space Caustics and Curvatures. In *Proc. of Pacific Graphics*, 181–188.

ZHOU, K., REN, Z., LIN, S., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Real-time Smoke Rendering using Compensated Ray Marching. *ACM Trans. Graph. 27*, 3, article 36.

ZÖCKLER, M., STALLING, D., AND HEGE, H.-C. 1996. Interactive Visualization of 3D-Vector Fields Using Illuminated Stream Lines. In *Proc. of IEEE Visualization*, 107–113.