Real-time Volume Caustics Rendering in Single-Scattering Media

Abstract

Volume caustics are beautiful patterns of light which are formed by light that first interacts with a specular surface and that is then scattered inside a participating medium. Although this phenomenon can be simulated, its generation process 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 idea of volume photon mapping and simulates volume photons along reflected and refracted rays. Instead of storing photons inside the volume, we directly display their contribution to the final image using a GPU-friendly ray marching algorithm. The implementation of our method is straightforward and we show that it can be seamlessly integrated in existing methods for rendering participating media. We achieve high-quality results at real-time framerates even for large and dynamic scenes. Furthermore, we show that our method is a good approximation of a ground truth volume photon mapping simulation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—

1. Introduction

Global illumination effects increase the realism of computer generated scenes. Especially caustics of reflective or refractive objects are visually pleasing. As can be seen in Figure ?? this is even better in participating media where volumetric caustics can be observed. However, most methods for volumetric illumination are slow, preventing interactive applications with such lighting effects. Therefore, we present a practical approximation of illumination inside participating media that runs in real-time. We do not need any precomputation, so any kind of dynamic objects can be illuminated. Since large and complex scenes are supported, our method is interesting for computer games. On the other hand, our method can be used as a fast preview generation for a more
complex simulation of illumination, as required e.g. for feature films.

Our main contributions are:

1. A real-time caustic volume method that seamlessly integrates into existing real-time volume rendering methods
2. A derivation of the simplifications of our method and a comparison with a ground truth simulation

The remainder of this paper is organized as follows: First, we review previous work in Section 2. Then Section 3 gives a short overview of our method. Afterwards, we derive the theory required for our method in Section 4. The implementation is then described in Section 5. In Section 6 we present our results before we conclude in Section 7.

2. Related Work

Since volume caustics are a kind of three-dimensional caustics generated inside participating media, the generation method is related both to the generation of surface caustics as well as to the rendering of participating media.

2.1. Surface Caustic Rendering

Surface caustics are usually generated by rays which refocus on diffuse surface after refraction or reflection on a specular surface. In computer graphics, photon mapping [Jen01] is the best method to successfully simulate this kind of phenomenon. Relying on the evolving GPU, several methods were developed to approximate real-time reflections [EMDT06] [USPK] [YYM05] and real-time refractions [OB07] [DW07] [Wym05], that allow fast photon path calculations on graphics hardware.

The first GPU photon map implementation was developed for participating media by Purcell et al. [PDC*03]. Recently, several methods for approximating caustics mapping have been proposed: [SKP07] [HQ07] [WD06] [SKALP05]. As in normal photon mapping, the methods follow specularly reflected and refracted photons backwards from the light are store their hit positions into a photon buffer. A second pass then reorganizes and splits these stored photons into a caustic map, which is projected onto the scene similar to a shadow map. In [Wym08] [WN], the speed and quality of this caustics mapping was improved. Jiongi et al. presented real-time caustics, computed in image space [LYL07]. Krueger et al. proposed a kind of screen-based photon ray tracing method for rendering surface caustics [KBW06].

2.2. Participating Media

Accurate rendering of volume caustics requires a simulation of single and multiple scattering in participating media [KVVH84]. A complete description of methods for rendering participating media is beyond the scope of this paper, a good overview can be found in [CPCP*05]. In this paper, our formulation is based on Nishita’s scattering equation [NMN87].

Typical applications for participating media are the real-time rendering of clouds [Harris01] [Dobashi2000] and smoke [RZLG08] [ZRL*08]. Approximating the shafts of light, which are typical effect of single scattering, can be achieved by blending layered materials [DYN02] or by warping volumes [IDN02].

Sun et al. proposed an analytical airlight model for real-time single-scattering in homogeneous media [SRNN05]. This work was extended by Wyman et al. to a real-time method for volume shadow rendering [WR08] Our volume caustics are an extension of this method.

2.3. Volume Caustics

Volume caustics can be computed with volume photon mapping technique [JC98] by tracing and storing photons inside the volume. Typical volume caustics are formed by shafts of light under water, as shown by Nishita et al. [NN94] and by Ernst et al. [EAMJ05] using a triangle-based caustics volume reconstruction method.

The eikonal rendering technique proposed by Ihrke et al. [IZT*07] is able to render all of the effects of refraction and participating media in real time. However, if there is any change to the lighting, materials, or geometry of the scene, several seconds are required for recalculating the radiance distribution. The method by Sun et al. [SZS*08] overcomes these limitations and achieves fully dynamic rendering of all the effects of refraction and single-scattering in participating media at interactive framerates.

In [KBW06], they also proposed to simply blend all the photon rays to coarsely approximate the volume caustics effects. Our method also rely on photon ray and share some similarities with this method. However, we create our photon ray in geometry shader and take the photon ray as a kind of intermediate primitive for quickly generating volume photons. Then in pixel shader we will precisely compute the scattered radiance on the whole transportation path from the light source to the view point. Complying with ray marching theory, in the final step we smartly blend the contribution of each photons for each view ray. Furthermore, we solved the visibility problem for caustics volume and we demonstrate that our method achieves similar results to volume photon mapping.

3. Overview

An overview of our method is shown in Figure 2. First, we render the airlight and the volumetric shadows [WR08]. The image with the volume caustics in then generated in the next render pass by computing the light paths from the light source which are reflected or refracted at a specular object. We draw these paths as lines in combination with a GPU
ray marching algorithm. This rendering pass is described in detail in the following sections. Afterward, we generate an image of the surface illumination and the surface caustics [1]. The final image is then composited of these three images. TODO: generate new images, airlight upper left, caustics upper right, surface illumination bottom left and final rendering bottom right.

Figure 2: The final image (upper left) is composed of an airlight image with a shadow volume (upper right) and a volume caustic image (lower left). The image on the bottom right shows the illumination on the surface.

4. Volume Rendering

In case of a participating medium, the light that reaches the eye does not emerge from a single direction due to scattering inside the volume. As shown in Figure 3, the viewer does not only see the radiance on the surface at point \( p \) which is attenuated due to absorption and out-scattering, but also some light which is in-scattered in the viewing direction at some point \( x \). A full computation of all such scattering events is called multiple scattering and very time consuming. Therefore a common approximation is the single scattering where at most one scattering event is assumed. In the following sections we will describe the single scattering and how volume caustics can be described.

4.1. Single-scattering media

In the following, we use the notation of Jensen and Christensen [1]. The medium is described by an absorption coefficient \( \alpha(x) \) and a scattering coefficient \( \sigma(x) \) for each point \( x \) inside the volume. The extinction coefficient is \( \kappa(x) = \alpha(x) + \sigma(x) \). To simplify the notation, we define transmittance \( \tau_{ab} = \exp(-\int_a^b \kappa(x)dx) \), which models the radiance reduction due to extinction (including absorption and out-scattering) from point \( a \) to \( b \). The in-scattered radiance \( L_i \) at a point \( x \) in direction \( \omega \) can be described as

\[
L_i(x, \omega) = \int_{\Omega} f(x, \omega', \omega) L(x, \omega') d\omega'
\] (1)

where \( f \) is the phase function that transforms the incoming radiance from direction \( \omega' \) to outgoing radiance in direction \( \omega \). Ignoring any emission inside the medium, the visible radiance for a viewer at position \( v \), looking in direction \( \omega \) can then be described as

\[
L_i(v, \omega) = L_A(v, \omega) + L_{IS}(v, \omega)
\]

\[
= \tau_p(v, L_i(p, \omega) + \int_p \tau(x, v) \sigma(x) L_i(x, \omega) dx
\] (2)

The first term \( L_A \) describes the attenuated radiance of a surface point \( p \) due to extinction inside the medium. The second term \( L_{IS} \) describes the in-scattered radiance along the viewing ray. Figure 3 visualizes this.

We assume the medium to be single scattering, such that radiance reaching the viewer has undergone at most one scattering interaction in the medium. Furthermore, the scattering is isotropic.

4.2. Volume Caustics

After discussing the single scattering we now describe the more general case where a refractive object is placed inside the medium between a light source \( s \) and the viewing ray \( pv \), as shown in Fig. 4. Since \( L_A \) is not affected by the occluder, only \( L_{IS} \) has to be recomputed. Due to the object the incoming rays will be refracted and leave the object in a different direction. Typically, this results in a shadow volume behind the object, because no direct light is arriving here. Instead, the radiance in other regions increases because the refracted rays concentrate here and form a volume caustic.
4.3. Volumetric Photon Mapping

In [JC98], volumetric photon mapping was introduced as a method to display volume caustics. Here, photons are first emitted from the light and then scattered and stored inside the medium. After each interaction with the medium, the photon changes its direction, simulating multiple scattering. These volume photons can then be used to display the illuminated volume using ray marching. Since this method is based on a traditional ray tracing framework its performance is too slow for real-time applications.

However, if we consider only single scattering, our scenario will be simpler. Here we assume that the photons travel on straight lines before entering the object and after leaving the refractive object. The change in photon direction is only caused by the refraction of the object. In this way we can treat the refracted rays as a kind of intermediate primitive for generating our volume photons. Using this simplification allows a GPU-friendly implementation as described in Sec. 5.

4.3.1. Displaying Volume Caustics

To display a volume caustic, the in-scattered radiance has to be split into two parts [1]: \( L_i(x, w) = L_{i,d}(x, w) + L_{i,i}(x, w) \), where \( L_{i,d} \) is the in-scattered radiance due to the direct radiance reaching \( x \) from the light source and \( L_{i,i} \) is the in-scattered radiance at \( x \) which arises from light which is refracted inside the object. The in-scattered radiance can then be described as: TODO: check if sigma/kappa is necessary

\[
L_{i}(x, w) = \int_{\mathcal{V}} \tau(x, v) \, L_{i,d}(x, w) \, dv + \int_{\mathcal{V}} \tau(x, v) \, \frac{\sigma(x)}{\kappa(x)} \, L_{i,i}(x, w) \, dv
\]

\[ (3) \]

The first integral describes the direct light contribution and we compute it using the method of Wyman et al. [2]. The second integral describes the in-scattered radiance of light which was reflected or refracted in the object. Note that the original formulation of [JC98] also used this separation, but here \( L_{i,i} \) was used for arbitrary multiple scattering inside the volume, whereas we only consider the refraction inside the object.

4.3.2. Ray Marching

To describe the in-scattered radiance at a point \( x \) in direction \( w \) we use the Volumetric Photon Map description [JC98]:

\[
L_{i}(x, w) = \frac{d^3 \Phi(x, w)}{\sigma(x) dV d\Omega} = \frac{1}{\sigma(x)} \sum_{p=1}^{N} f(x, \omega_p, w) \frac{\Delta \Phi_{p,i}(x, \omega_p)}{\Delta V(x)}
\]

\[ (4) \]

Here, the radiance is computed by collecting all volume photons in a small volume \( \Delta V \) around \( x \). Each incoming photon carries a radiant flux \( \Delta \Phi_{p,i} \), which is then weighted with the phase function \( f \) and summed. To compute the flux \( \Delta \Phi_{p,i} \) of the incoming photon we assume that the total flux \( \Phi \) of the light source is subdivided into the flux for \( N \) photons: \( \Phi = \sum_{p=1}^{N} \Delta \Phi_{p,i} \). If we consider the absorption and out-scattering along the photon path from the source \( s \) to \( x \), the incoming flux is

\[
\Delta \Phi_{p,i}(x) = \Delta \Phi_{p,s}(x) \tau(x, \omega_p) \tau(p_1, x)
\]

\[ (5) \]

\( \Delta V \) is a small search region where the surrounding photons of \( x \) are located. Often, a sphere is used as a search region. Using ray marching with a fixed step size \( \Delta x \), the volume caustic radiance can be approximated as a sum with \( M \) steps

\[
L_{CV}(x, w) \approx \Delta x \cdot \sum_{i=1}^{M} \tau(x, v) \, \frac{\sigma(x)}{\kappa(x)} \, L_{i}(x, w)
\]

\[ (6) \]

Where the positions along the viewing ray are computed as \( x_i = v + i \cdot \Delta x \). Using the definition for \( L_{i,j} \) (Eq. 4), this results in

\[
L_{CV}(x, v) \approx \Delta x \cdot \sum_{i=1}^{M} \tau(x, v) \, \frac{\sigma(x)}{\kappa(x)} \, L_{i}(x, w)
\]

\[ (7) \]

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

5. Screen-based volume caustics

The basic idea of our method is to simulate volume photons along photon rays to display approximate volume caustics in real-time. We therefore generate the points of the photon path on the GPU using a fast reflection/refraction method. Next, we connect the points with lines and draw the lines in...
combination with a GPU ray-marching algorithm to display the caustic radiance. We will explain each step separately in the following sections.

5.1. Generate reflected/refracted photon ray

To simulate the light transport through a refractive object we compute the refracted ray when it enters the object and then the refracted ray when it leaves the object. In our implementation, we use the method by Wei et al. [HQ07] to efficiently compute these two points and directions. For efficiency reasons, we ignore further reflections or refractions, a discussion about this can be found in Sec. ...

Based on Fig. 6 we summarize how to trace the photon rays for two-bounce refraction:

1. First we render the whole scene except of the refractors into a background texture which also includes the depth values.
2. Then we switch the view to the light source and render the reflective objects. The closest and most distant point are stored in a texture.
3. Next, we start tracing [HQ07] and compute the position \( p_1 \) where the light left the refractors and the reflected/refracted ray direction \( v_1 \) (see Fig. 6).
4. Based on \( p_1 \) and \( v_1 \), we traverse the depth values of the background texture to compute the intersection position \( p_2 \) of the refracted ray and the scene.

Figure 5 (left) shows the computed exit points \( p_1 \) and surface points \( p_2 \) for a glass sphere above a ground plane. // as well the connected lines. textbfTODO: new image

5.2. Ray Marching

The illuminated volume can be displayed by ray marching. Here, we step along the view ray from front to back and sum the radiances contribution at each point, taking the attenuation to viewer into account. This requires the storage of the density values in a volumetric data structure, like a 3D grid. For an efficient GPU implementation, we use a different method for ray marching through the volume which is visualized in Figure 6. Instead of pre-computing and storing the density values, we draw the photon paths as lines and compute the contribution for each fragment during rasterization. This is implemented as follows: After computing the points \( p_1 \) and \( p_2 \) for all photons, we draw a line for each pair of points \( (p_1, p_2) \). The camera is set to the viewpoint and a fragment program computes the 3D position \( x \) for each pixel of the line during rasterization. Since our goal is to compute Eq. 7 for the pixel, we first calculate the distances \( d_{p_1}, d_{p_2}, d_{sv} \) that we need to compute the corresponding transmittances \( \tau(x, p_1), \tau(p_1, x), \tau(x, v) \). As a search volume \( \Delta V \) we use a small cuboid which is oriented along the viewing ray, as shown in Figure 6. The extents of the cuboid are defined by the pixel area \( A_{\text{Pixel}} \) and a user-defined search range \( r \) along the viewing ray (so the volume is \( \Delta V = 2rA_{\text{Pixel}} \)). This allows us to compute the contribution of one photon to the caustic radiance in Equation 7. If we repeat this process for all lines and accumulate the results, we compute the volume caustic radiance by ray casting.

If the search region \( r \) is smaller than the step size \( \Delta x \), it is possible that we have to ignore a fragment if the 3D position is outside the search region. Therefore, we compute the relative position of \( x \) in the bin and discard the fragment if the distance is larger than \( r \). In this way, we can use a GPU-friendly ray-marching that does not require any 3D storage or sorting from front to back. Moreover, we do never step in empty regions.

Figure 6: Ray Marching is implemented by drawing lines from \( p_1 \) to \( p_2 \) (red). For each pixel of the rasterized line, the contribution of a photon, located at \( x \) is accumulated. The attenuation is computed along the whole path from \( s \) to \( v \), shown in blue. The ray marching steps are shown for only one pixel for visualization.

5.3. Airlight and Surface Illumination

To compute the airlight and volumetric shadow, we use the method proposed by Wyman et al. ?? The surface illumination is computed with Hierarchical Cautics Mapping ??, based on the attenuated points \( p_2 \) on the surface. (what else?)
5.4. Compositing the Final Image

As shown in Figure 2, the final image consists of three components: the airlight image, the volume caustic image, and the background image. Since light is additive, we can simply add our volume caustic image to the airlight image. To correctly display the background image, we compute the attenuation $\tau(p, v)$ for each pixel position $p$ to the viewpoint and use this as a scaling factor for the background radiance before adding it to the final result (Eq. ??).

5.5. Discussion

Our method simulates the most important light paths. However, several things can be added: First, we only consider the refracted ray at the points $p_0$ and $p_1$ to display the volume caustics. A complete simulation would also include the reflected ray which could be implemented as a multi-pass algorithm for multiple reflections and refractions. Fresnel reflectances as well as possible absorption inside the refractive object could easily be included.

TODO: mention the inhom. medium somewhere

Do we have precision problems if the viewer is looking approximately along the photon ray (line with only few pixels)?

Create some animation where the viewer is moving from outside to inside the caustics volume.

Do we start to see the individual lines if the viewer is moving very close (or inside) the volume caustic? Do we need a Gaussian blur in image space?

TODO: generate a video where the camera is flying from outside -> inside -> outside the caustic of a sphere

6. Results

To verify the accuracy of our method, we compared our volume caustics with a ground truth image generated with Mental Ray (see Figure 7). Note the very similar patterns of light which are generated with both methods. However, we can observe several differences: Since the Mental Ray image is based on volume photon mapping, multiple scattering is simulated and the overall appearance of the streaks of light is a bit more blurred. Additionally, we can not simulate all light paths, e.g. the refraction on the foot of the bird. For more complex objects, like the dragon, the appearance of the caustics is slightly different since we assume only two refractions for each ray. While the volume photon mapping took ... to ... minutes, of method runs with 20 - 30 fps. TODO: fill correct timing values here TODO: try the dragon as mental ray comparison with two different illumination directions.

Figure ... shows how the image quality and framerate changes when changing the number of photon rays. Additionally, we show how the stepsize for ray marching affects the image quality.

A strength of our method is the temporal coherence: As shown in the accompanying video, animated and deforming objects can be displayed correctly without flickering. We can treat arbitrary deformations without any precomputation. Figure 8 shows some examples of animated scenes.

Figure 7: Comparison of volume photon mapping (left) with our method (right).

Figure 8: Our method displays arbitrary deformations without precomputation.

Although the airlight illumination assumes a homogeneous medium, our volume caustics can be generated for inhomogeneous media, as shown in Figure 9.

submitted to Pacific Graphics (2009)
TO DO: generate images with varying number of photons and varying stepsize for ray marching, show how fps changes

TO DO: generate comparison images for simple blending vs. ray marching, show blending is too bright

TO DO: one idea for an image: place the ring under water, show god-rays and volume caustics

TO DO: idea: show some dispersion effects, e.g. from a diamond. Use white input light and compute three volume caustics for red, green, and blue, each with a slightly different index of refraction

7. Conclusion and Future Work

We presented a real-time method for rendering caustics in volumes. Based on existing methods for rendering airlight and volumetric shadows, we extended this to volumetric caustics, allowing the display of the most important volumetric effects in real time. Our method is an approximation for single scattering. Since it works for large and dynamic scenes we believe that it can be applied in computer games as well as a preview for a more complicated lighting solution.

As future work we consider to compute multiple reflections and refractions using depth peeling in combination with multi pass rendering. This enables a more correct solution. Another possible extension would be the display of dispersion effects by rendering multiple passes with different color bands and varying refraction values. Moreover, we think about computing volumetric effects in inhomogeneous media where light travels along curved rays, like the eikonal rendering.

References


