2011 12th International Conference on Computer-Aided Design and Computer Graphics

Real-time Volume Caustics with Image-based Photon Tracing

Yuxiang Wang* † ‡ §, Bin Wang* ‡ § and Li Chen* ‡ §
*School of Software, Tsinghua University, China
†Department of Computer Science and Technology, Tsinghua University, China
‡Key Laboratory for Information System Security, Ministry of Education of China, China
§Tsinghua National Laboratory for Information Science and Technology, China

Abstract

Rendering of volume caustics in participating media is often expensive, even with different acceleration approaches. Basic volume photon tracing is used to render such effects, but rather slow due to its massive quantity of photons to be traced. In this paper we present an image-based volume photon tracing method for rendering volume caustics at real-time frame rates. Motivated by multi-image based photon tracing, our technique uses multiple depth maps to accelerate the intersection test procedure, achieving a plausible and fast rendering of volume caustics. Each photon dynamically selects the depth map layer for intersection test, and the test converges to an approximate solution using image space methods in a few recursions. This allows us to compute photon distribution in participating media while avoiding massive computation on accurate intersection tests with scene geometry. We demonstrate that our technique, combined with photon splatting techniques , is able to render volume caustics caused by multiple refractions.

1. Introduction

Global illumination contributes a lot to the synthesis of photorealistic images, including light paths interacting with surfaces and participating media. Caustics caused by scattering, refraction or reflection of light are stunning visual effects, yet difficult to render using off-line approaches, and even harder for volume caustics in participating media.

Different approaches, such as path tracing [1] or volume photon tracing [2], are used for the rendering of volume caustics, but too expensive for interactive rendering as they need a large number of paths or photons to be traced to avoid undersampling. Spatial traversal structures [3] [4] are used to accelerate the rendering process. However, precomputation is needed for those structures, and they are also hard to implement on GPU. Recent methods [5] [6] [7] [8] trace beam instead of photon or ray to achieve interactive speed. However, to obtain better result, these methods require massive number of beams, and high fillrate to render them.

In this paper we present a novel scalable method on rendering volume caustics in single-scattering participating media. We use multiple depth maps for volume photon tracing and photon splatting for radiance estimation. Instead of tracing photon in participating media accurately by doing intersection tests with scene geometry, we estimate the intersection point in image space. The images used for intersection estimation are generated using depth peeling method in order to present more details of the scene geometry, even for the hidden faces. With these images, we propose a robust image-based method for photon-geometry intersection which can converge to an approximate solution fast.

Our method is briefly described in Figure 1. On a coarse level, our method can be divided into four steps. First, we render multi-layer depth map and normal map as clues for intersection test(A). Maps that have different distances from the light are rendered on separate layers, in order to both front and back faces' information. Second, photons are traced in image space using multiple depth and normal maps(B), with the method described in Section 4. Since volume caustic is generated mostly after two refractions on specular objects, each photon is tested multiple times, with at least two layers of the depth map, to construct the whole light path that formulate volume caustic. We store the photon position and flux into textures. Third, we render the photons as point primitives and splat them on screen, blending photons splatted at the same screen pixel as a gathering step(C). Volume fog [9] and direct lighting images are also generated. The final image is the combination of these three images(D). Section 4.3 has a more detailed description of the method.

Applying the light transport model in single-scattering participating media to the photon tracing procedure as well as the intersection estimation, our method can generate volume caustics at real-time speed without pre-computation, and the scene as well as light source can be changed dynamically. Due to its photon tracing basis, our method can be further extended to render other effects like volume shadow and multi-scattering effects. The method is simple and can be implemented on common GPU easily, and can work with other rendering methods, e.g. volume fog generation and direct lighting, as a complement.

978-0-7695-4497-7/11 $26.00 © 2011 IEEE
DOI 10.1109/CAD/Graphics.2011.24

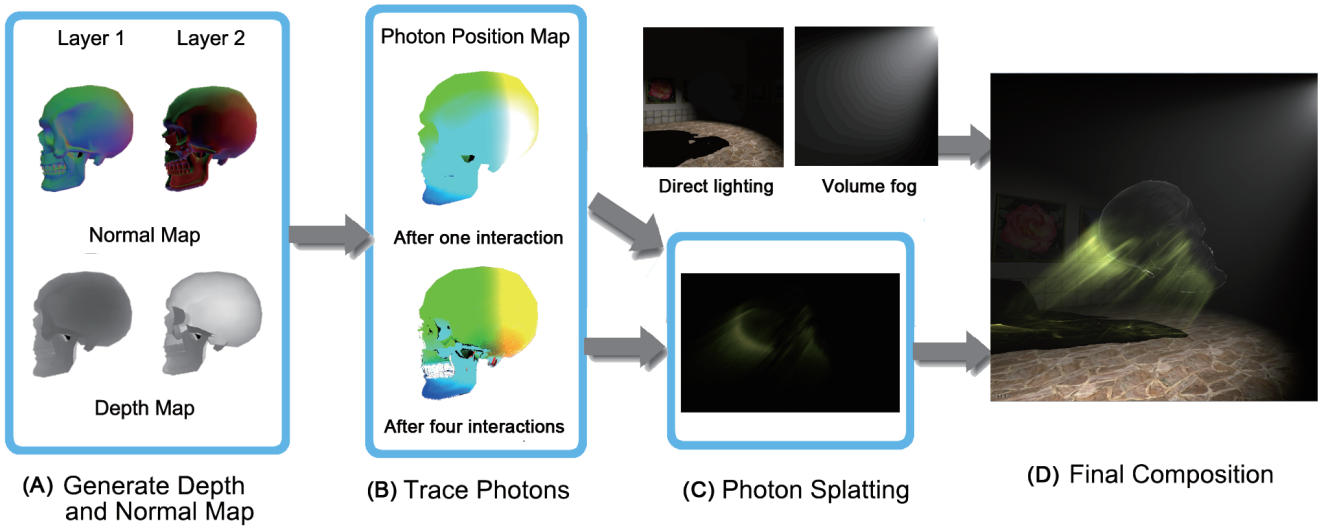

Figure 1. Step-by-step overview of our algorithm. (A) generates normal maps and depth maps from light's view. Photons are then traced, formulating photon maps where colors in the maps represent the photons' position, as shown in (B). Then the photons are splatted(C), and combined with other image components as final result(D).

2. Related Works

Caustics are formed by focusing or defocusing of light rays after they interact with specular surfaces. Surface caustics are generated when these light rays interact with diffuse surfaces, and volume caustics with participating media. Caustic rendering techniques can be categorized into these two related fields.

2.1. Surface Caustics

Surface caustics are light projected on diffuse surfaces after refraction or reflection on specular surfaces. Photon mapping [10] is often used as the ground truth of such visual effects. Purcell et al. [11] described the first GPU photon mapping algorithm. Wang et al. [3] introduced a GPU-based photon mapping method with final gathering, with the acceleration of GPU-based kd-tree [4]. It produces convincible results, but relies on sparse gathering location and complex acceleration structures.

Latter approaches mostly tune the two passes of basic photon mapping method: photon tracing pass and rendering pass, approximating the effect for speed-up. Szirmay-Kalos et al. [12] and Yao et al. [13] used image to trace rays after refraction, avoiding the massive photon-triangle intersection. Photon splatting technique is often used to approximate radiance in the rendering pass of photon mapping in interactive rendering, instead of nearest neighbor search. Caustic mapping [14] renders caustic images akin to a shadow map, and projected it onto scene during the rendering pass. It suffers from the noise in regions that receive few caustic photons when adopting photon splatting technique for caustic map generation. Wyman et al. [15] addressed this problem by using adaptive and hierarchical sampling with different splatting size. Yu et al. [16] presented a method based on geometric parametrization. It parameterizes the caustic rays with parallel planes, modeling the effects with general linear camera model.

2.2. Volume Caustics

Volume caustics can be observed when light rays propagate in participating media. The scattering and emission effects make light rays visible through the participating media. Jensen and Christensen presented the volume photon mapping method [2], which stores and traces photons not only on surfaces but also through the participating medium. This method is used as ground truth for visual effects such as volume caustics. Jarosz et al. [5] proposed beam tracing techniques to gather photons through a beam in photon gathering pass, avoiding ray marching procedure along the view rays to gather nearby photons. Recent approaches [7] [8] generated triangle-based beam for volume caustics, and achieve real-time frame rates.

Approximations of light transportation are commonly used in interactive rendering of volume caustics. Eikonal rendering technique [17], another interactive-speed rendering technique, stores flux to voxel generated by discretizing the scene, and computes flux transport using first-order approximation of eikonal equation. However, pre-computation is needed for this technique, so it cannot handle dynamic scenes. Sun et al. [18] combined photon mapping method with eikonal rendering, and used oct-tree to store photons,

achieving fully dynamic rendering of single-scattering effects at interactive frame rates.

Kruger et al. [19] introduced image-space caustics rendering with lines as querying primitives, which give cues to splat energy directly to screen pixels. Hu et al. [6] extended the line-rendering idea to treat lines as rendering primitives, and directly rendered light rays that form the volume caustics pattern, achieving real-time frame rates. Sun et al. [20] transformed light rays and view rays to another coordinate system, making ray-ray intersection test easier to compute and rendered volume caustics accurately. However, it is still an off-line method even with GPU acceleration. Photon splatting technique is also used for rendering effects in participating media [21], and we use a similar method to splat the traced photons to image.

2.3. Image-based Ray Tracing

Intersection estimation is often used in interactive rendering, instead of accurate intersection test. Wyman [22] rendered both front and back face of refractive objects, using light ray's incident direction and the outgoing directions after first refraction to interpolate its intersection point on backward faces. Hu et al. [23] improved this method by first calculating the face where the second intersection point lies on, then used ray marching technique to find the correct intersection point. Both techniques can handle refraction and reflection of one specular object, but no more than two bounces because of using only one texture.

For scenes with multiple specular objects, Szirmay-Kalos et al. [12] used environment maps as distance impostor, doing iterative refinement in image-space to get an approximate intersection between light ray and environment map, retrieving the background information of where the refracted light ray hits. However, it cannot find the correct solution when the desired intersection point is occluded or the desired point is not the first intersection point when looks from environment map center. Yao et al. [13] addressed this problem by using multiple environment maps that cover the scene adequately.

Most related to our work, Sheng et al. [24] rendered depth textures and did binary search in image space to find the intersection point. However, their binary search is done on orthogonal depth maps, which may lead to significant approximation error when the scene is viewed from a camera with a large field-of-view. Another binary search method in GPU Gems 3 [25] approximates intersection points using environment maps. Due to its environment map based nature, it cannot process some complicated situations efficiently, which we will discuss in Section 4.1.

3. Radiance Estimation using Volumetric Photon Splatting

In the presence of participating media, radiance undergoes three kinds of phenomena: absorption, scattering and emission. When light transports in the participating media, the radiance of point x follows the light transport equation:

$$L(x) = \underbrace{\tau(x_0, x)L(x_0)}_{L_{ri}(x)} + \underbrace{\int_{x_0}^{x} \tau(u, x)\kappa_t(u)J(u)du}_{L_m(x)}, \quad (1)$$

where $\tau(x_0, x)$ denotes the transmittance from point x_0 to x, $L_{ri}(x)$ denotes the incident radiance due to the point x_0 on background surfaces, $L_m(x)$ denotes the medium radiance contribution due to the source radiance within the medium, $\kappa_t(u)$ denotes the extinction coefficient due to absorption and out-scattering, and $J(x)$ denotes the radiance added to point x due to self-emission and in-scattering.

For rendering of volume caustics, we only consider single-scattering participating media, which describes haze but not thick fog and cloud. In single-scattering model, scattering is considered only once in participating media. Viewed from the eye, the radiance transferred to viewer at position v from direction ω_0 is given by

$$L_{eye}(v, \omega_o) = \int_v^{\infty} \tau(v, x)\kappa_t(x) \int_S p(\omega_o, \omega_i)L(x)d\omega_i dx, \quad (2)$$

where $p(\omega_o, \omega_i)$ denotes the phase function of incident direction ω_i and outgoing direction ω_o. Using photon to estimate volume radiance, $L(x)$ should be represented by flux using the following equation because photon represents incoming flux:

$$L(x) = \frac{d^2\Phi(x, \omega)}{\kappa_s(x)d\omega dV}, \quad (3)$$

where $\kappa_s(x)$ denotes the scattering coefficient and $\Phi(x, \omega_o)$ denotes flux of a photon at position x with direction ω. Thus Equation 2 can be converted to use nearest photon search:

$$L_{eye}(v, \omega_o) \approx \sum_{p=1}^{n} \tau(v, x)\kappa_t(x)p(\omega_o, \omega_i)\frac{\Delta\Phi_p(x, \omega_o)}{\frac{4}{3}\pi r^3\kappa_s(x)}. \quad (4)$$

While performing nearest neighbor search is costly on GPU, volumetric photon splatting [21] is another suitable solution. Splatting the volume photon to a sphere, Equation 4 can be written as:

$$L_{eye}(v, \omega_o) \approx \begin{array}{l} \sum_{p=1}^{n} \tau(v, x)\kappa_t(x)p(\omega_o, \omega_i)\times \\ \sum_{j=1}^{m} K(x_j - x_p)\Delta x_j \frac{\Delta\Phi_p(x_p, \omega_o)}{\frac{4}{3}\pi r^3\kappa_s(x)}. \end{array} \quad (5)$$

$K(x_j - x_p)$ denotes filter value at $x_j - x_p$ from kernel center, x_j denotes the intersection point between view ray and photon splat sphere, x_p denotes the position of photon, and Δx_j denotes the distance view ray advances inside the photon sphere. According to Equation 5, each visible photon will affect a certain set of screen pixel, and splat energy on these pixels directly without direct neighbor search.

Radiance of volume caustics can be easily computed by blending splatted fragments from different photons together, formulating the sum value in Equation 5. In our test, radiance estimation using photon splatting only takes 23% of the whole rendering time, but interaction test takes about 70%, which makes it the bottleneck of the whole rendering procedure. In the next section, we will describe how to estimate intersection easily on GPU using image-space method.

4. Screen-space Volume Photon Tracing

To acclerate the photon tracing procedure, the key problems is how to compute the intersection points of photons and scene geometry quickly. Here we present a multi-image based method of intersection estimation. We will first explain how it works with one depth image, and then extend it to multiple images that can represent the whole object. Finally we will present the screen-space volume photon tracing method using intersection estimation.

4.1. Intersection Estimation with Single Depth Image

When tracing photon in participating media, the photon interacts with both medium and scene geometry. At the time it hits a surface, the photon will be absorbed/reflected/refracted just like normal photon tracing; but it will also have chance to interact with medium, and be absorbed or scattered. The average distance that a photon can advance between two interactions with medium follows the next expression:

$$d = -\frac{\log \xi}{\kappa_t}, \tag{6}$$

where $\xi \in (0, 1]$ is a random number uniformly distributed in the range. This parameter affects the average length of caustic light beams. The photon may hit specular surfaces in that distance, so we need to detect the intersection within the line segment $l = x_{start} + t \cdot \omega$ that is determined by d, the photon's direction ω and photon's last position x_{start}.

We render the depth map from the light source as a texture, with each pixel storing the distance to the light source. By projecting l to the depth map, depth values of x_{start} and photon's end position $x_{end} = x_{start} + d \cdot \omega$ recorded in the texture can be retrieved from the depth map, denoted as d_s and d_e. At the same time, the real distance

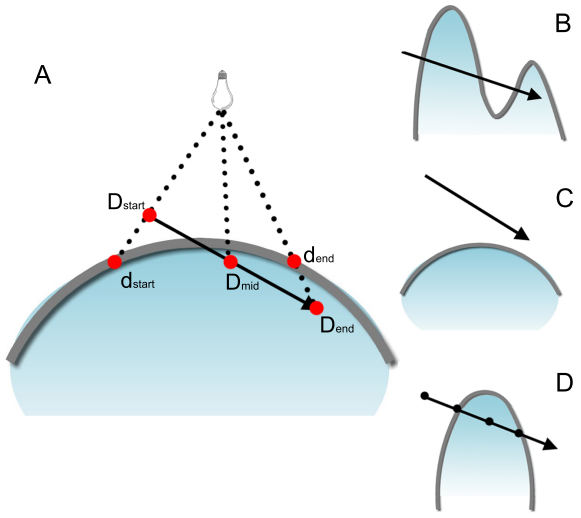

Figure 2. Image-space intersection estimation. (A) is the normal intersection situation. $[D_{start}, D_{mid}]$ is selected for next iteration. (B) has multiple intersection points; (C) has no intersection with surface. (D) has a "peek" in the route, and can be up-sampled like the black dots indicated.

of x_{start} and x_{end} from the light source can be directly computed, denoted as D_s and D_e.

When $d_s > D_s$, it means that x_{start} is closer to the light source than its projection on the texture, and vice versa. Compare two pairs of depth values of x_{start} and x_{end}, one from the depth map(d_{start} and d_{end}) and the other is the real distance computed from coordinates(D_{start}, D_{mid} and D_{end}). If the relationship of these pairs differs, e.g. $d_s > D_s$ and $d_e < D_e$, it means that there is an intersection between x_{start} and x_{end}. Then select the middle point x_{mid} of l. Repeat the aforementioned procedure to determine whether x_{mid} is below the surface; if so, then recursively find the intersection point in $[x_{mid}, x_{end}]$, otherwise in $[x_{start}, x_{mid}]$.

There exists some special situations, shown as B, C and D in Figure 2. If the relationship of depth value pairs differs as aforementioned, then the intersection must be type A or B. The difference between A and B is that B has multiple intersection points in the route, and we only need the first intersection point. To make the final solution converge to the first intersection point from x_{start}, $[x_{start}, x_{mid}]$ is chosen in possible situations, rather than the other half, for further iteration to find the intersection point .

Intersection type C has no intersection point with surfaces, so it will not have the "different" relationship aforementioned until the recursion stops, and detects no intersection. D has a "peak" of geometry inside the route, and the behavior for intersection test is much like type C, which

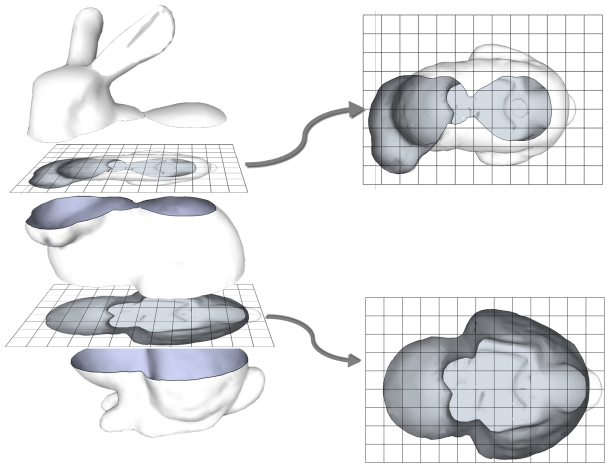

Figure 3. Depth map rendering. The depth maps are rendered at different depth value, and each layer contains the geometry information under this threshold. For each depth map on the right, the darker part indicates the back-facing part of model(e.g. interior part of the model), and the brighter part indicates the front-facing part(e.g. exterior part of the model).

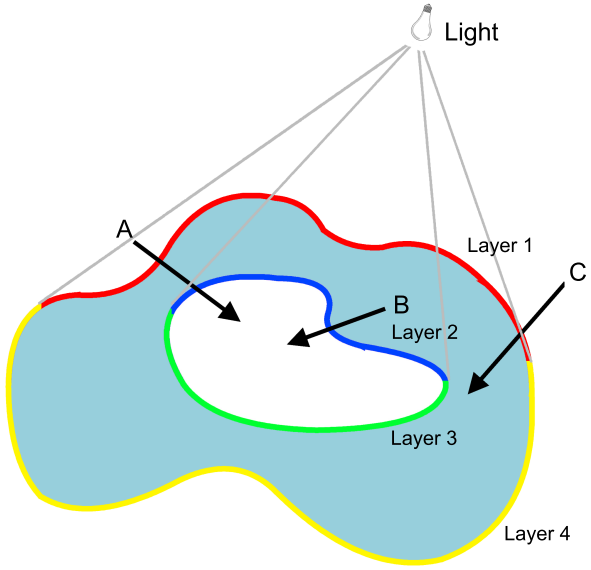

Figure 4. Intersection estimation with multiple depth maps. Different color indicates surfaces represent by different depth layers.

makes it complicated to deal with. For both type C and D intersections, we sample uniformly on the route to see if all the samples are below/above the surface. The sampling only needs once at the beginning of the line's intersection detection, and it will add few costs to the recursion since few samples are needed, and if the samples are all below/above the surface, further iterations are canceled. In our experiments, 30 iterations of binary search and 8 samples of "peak" detection are enough for intersection estimation, which is fast with GPU acceleration.

Compared with binary search method proposed by Umenhoffer et al. [25], our method is more efficient and robust. Umenhoffer's method, which estimates ray ends using scene information and binary search for the intersection point using environment maps, cannot handle intersection type C and D and lead to error when used in participating media. An intersection point is always calculated using Umenhoffer's method, which is satisfied by its environment map based nature. However, intersection point does not always exist in participating media(Type C), and Umenhoffer's method still searches for intersection point along an entire ray that ends at the boundary of the scene for type C, which is unnecessary and inefficient. Type D is based on type C(has no intersection on initial detection), and cannot be handled either.

4.2. Intersection Estimation with Multiple Depth Images

Specular objects may have different surfaces that will interact with photons, e.g. front face and back face. One depth image cannot hold information for a complete photon route, so we use a method similar to depth peeling [26] to generate multiple depth maps for intersection estimation. Figure 3 shows the image generation procedure. For a certain level of depth map, only fragments whose depth values are greater than a threshold are rendered, and for each threshold, front facing meshes and back facing meshes are rendered in two passes. Bounding box of the specular objects is divided uniformly on the light source's spot direction, to select the depth thresholds.

We perform the single-image estimation for each level of the depth map, recording the line coefficient t for each level that intersects with the photon, and select the smallest t to compute the intersection point. That will ensure the algorithm finds the first intersection at the right level. For path A in Figure 4, the algorithm will find intersection points on both Layer 1 and Layer 2, but comparing t for these two intersections, the one on Layer 2 is discarded. For path B, intersection estimation on Layer 1 will find no intersection and converge to x_{start}, because both x_{start} and x_{end} are under the surface; but on Layer 2 it will find the intersection. And for path C, the tests on Layer 2 and deeper will find no intersection for the same reason. So the layer selection can be done dynamically in image space.

4.3. Volume Photon Tracing with Intersection Estimation

With the intersection estimation method, we present the screen-space volume photon tracing algorithm.

First we render depth maps with normal information from the view of light source, and store them in a multi-layered texture(Figure 1(A)). Then, photons are emitted from the light source by drawing specular objects to textures, and every pixel drawn represents a photon, formulating the photon maps. The photon maps are square textures, and each pixel of the photon map records the position, direction and power of a photon. The upper picture of Figure 1(B) shows the position information of the initial photon map. Each photon map contains photon information after one interaction, so multiple photon maps are needed which differ from the traditional photon mapping.

Next, we trace each photon with a user-defined depth of path. In one trace procedure, we detect the first hit point on specular surfaces. If the light ray hits the surface, then record the hit status and find the next intersection point, so these two points define a path that crosses the specular object(or part of it, e.g. into a hole inside the object). If it does not hit the surface, the photon will proceed along its previous direction, and photon power attenuates according to light transport equation. New photon positions are saved for next trace procedure. When the photon hits a surface, the direction is changed according to the normal direction of surface at the hit point, and photon power is also changed due to surface interaction and BRDF. After several interactions, photons are not uniformly distributed along the surface, but scattered to many non-related directions, which can be revealed from the noise on position photon map. The lower picture of Figure 1(B) shows position information of photons after 4 interactions. For the following procedure, we read the photon information from photon maps and use it to trace the next interaction.

Finally, we read the position and power from photon maps and render photons at the correct positions in the world space as point primitives using this information, then splat the point using a Gaussian filter(Figure 1(C)). When splatting a photon, the distance between the photon and eye's position is taken into consideration, in order to simulate the attenuation along the view ray. Different interaction steps are splatted separately, e.g. the upper of Figure 1(C) is splatted photons after the first interaction and the lower one is splatted after four interactions with the participating media. Those images representing individual interaction steps are combined together for the final image. Different point primitives are blended together, so caustic light beams will look brighter because intensive photons lie inside the beams. Computing kernel functions on the fly, instead of using precomputed textures, makes photon splatting faster. The splatting procedure results in a texture that contains the volume caustic pattern in eye's view. The volume fog texture, background and specular objects are rendered using different methods, and then final image is the sum of those three textures and the volume caustic pattern.

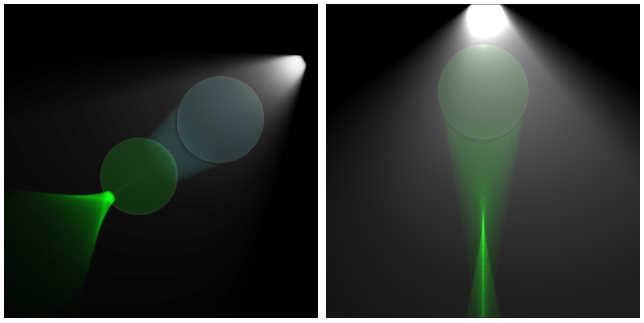

Figure 5. (left)Light propagates through two specular objects. Both light color and direction change due to multiple refractions. (right)Light refocuses after two refractions with the glass sphere.

Table 1. Scene information and framerates.

Scene	Sphere 1	Sphere 2	Bunny	Feline	Skull
Triangle #	40k	20k	10k	150k	10k
FPS	23.5	23.8	22.6	20.5	25.0

5. Implementation

Our method is implemented with OpenGL 4.0 and GLSL 1.5, and the algorithm runs on an nVidia GeForce GTX 480 graphic card, Intel Core i7 x870 with 8 Giga bytes RAM.

The photon map consists 3 different textures to contain position, direction and power. An additional texture is used to save photon status, e.g. whether it needs to be traced in the next pass. Framebuffers are used to render on textures. Unlike the traditional photon mapping, only caustic photons are generated, and they are emitted towards the specular objects. In the final rendering pass, all photon point primitives are generated using geometry shader, and are directly splatted in fragment shader. Photons inside the specular objects are omitted, because reflection and attenuation inside the specular object dominate the visual effect.

The direct lighting is rendered using shaders according to Phong model, and volume fog effect is rendered using method addressed in [9].

6. Results and Discussion

All the test scenes are rendered at 800×800 pixels, and the resolution of photon map for each bounce is 1500×1500. We use 3-4 layers of depth map and 3-4 bounces for each photon. More depth layer can be added for complex scenes. The scene information and frame rates are shown in Table 1. Light direction and intensity, participating media parameters and object can be dynamically changed in scene, and photons are retraced in every frame.

Figure 5 shows the interaction of light with spheres, which is easier to imagine the actual light paths. For the left image,

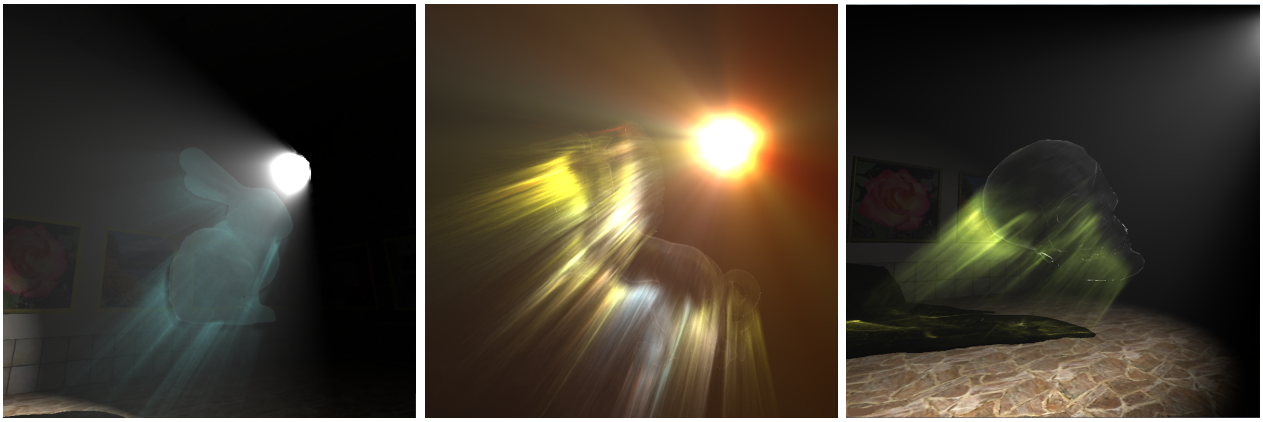

Figure 6. Real-time volume caustics caused by specular objects.

since multiple layers of depth map are used, we can find the correct light paths even the green sphere is totally blocked viewed from the light source. Light changes route after the interaction with the green sphere, and changes color using the color and normal information stored in the deeper level of the depth map. For the right image, we can clearly observe that light reconverges after two refractions with the sphere, forming a bright line in the caustic pattern.

Figure 6 shows three scenes with different meshes. Caustic are generated after refractions, making bright light beams prorogate to different directions.

Our algorithm is suitable for scenes with multiple specular objects, which cannot be handled well using other real-time approaches. Since intersection estimation takes little time even with more depth map layers, it is easy to find all hidden specular surfaces by adding more layers to the depth map.

Another ideal scenario of our algorithm is that the light source is close to the specular objects, which can make full use of the photon map. If light source is far away from the objects, the specular objects will look small from light source and valid photon count decreases.

Here we compare our method with other momentous approaches on volume caustic rendering in the past few years. Sun et al. [18] used photon tracing to find light paths through specular objects and ray marching for rendering, achieving interactive frame rates of 3-5 Hz. Nevertheless, an oct-tree is constructed for the scene geometry on GPU, which leads to complexity in implementation.

Line space approach [20] transforms light and view rays to Plucker coordinate system, making ray-ray intersection test easier by simply computing a determinant value. This method gives accurate rendering result, but it takes several minutes to render even with nVidia CUDA acceleration, and a bounding box hierarchy is also constructed on GPU to speedup the computation. The biggest advantage of our method comparing with these two approaches aforementioned is that our implementation needs no acceleration structure, and is much easier to implement without using advanced graphic hardware.

Hu et al. [6] rendered light rays as line primitives to form volume caustic pattern. They used textures to store photons and depth maps to find light paths in the similar way as ours, rendering volume caustics at 11-30 Hz. Only one depth map is used for intersection estimation in their method, so it cannot trace rays that carry out multiple refractions. Additionally, it is hard to use in inhomogeneous media, where different lines need to be connected to form a curved light path, while our photon tracing based method can handle easily with small tweak, because a series of photons on the light path directly forms the curved path.

7. Conclusion and Future Work

In this paper we proposed a robust method to compute volume caustics in image-space at real time frame rates. Traditional photon mapping methods are modified to use multiple depth maps as distance impostor, and an intersection estimation method using the depth maps is proposed to speedup the photon-geometry intersection test. Point primitives are directly rendered on screen as photons, using photon splatting approaches. Unlike many other real-time volume caustic rendering approaches, we can correctly handle multiple specular objects with occlusion, and give plausible rendering results.

Based on photon mapping, our method still relies on increasing photon counts and improving the gathering technique to eliminate noise. Using more photons means larger textures, and results in lower speed. As future work, adaptive splatting of the volume photons can be used to decrease the photons needed and further improve the results.

Another interesting direction of future work is on sampling. We sample photons uniformly in the view of light source, but it suffers under-sampling when light is far away from the object. When there are sharp features on the

surfaces, they need more photons because sharp curvatures are likely to produce visible caustic beams, which cannot be satisfied by uniform sampling. Adaptive sampling of photons can further enhance the algorithm, making rendering speed and visual effect better.

Acknowledgment

The research was supported by Chinese 973 Program (2010CB328001), the National Science Foundation of China (61003096, 60773143) , and the NSFC-JST Key Joint Funding Project(51021140004).

References

[1] E. Veach, "Robust monte carlo methods for light transport simulation," Ph.D. dissertation, Stanford University, 1997.

[2] H. W. Jensen and P. H. Christensen, "Efficient simulation of light transport in scences with participating media using photon maps," in *SIGGRAPH '98*, 1998, pp. 311–320.

[3] R. Wang, R. Wang, K. Zhou, M. Pan, and H. Bao, "An efficient gpu-based approach for interactive global illumination," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 91:1–91:8, July 2009.

[4] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time kd-tree construction on graphics hardware," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 126:1–126:11, December 2008.

[5] W. Jarosz, M. Zwicker, and H. W. Jensen, "The beam radiance estimate for volumetric photon mapping," *Computer Graphics Forum*, vol. 27, no. 2, pp. 557–566, 2008.

[6] W. Hu, Z. Dong, I. Ihrke, T. Grosch, G. Yuan, and H.-P. Seidel, "Interactive volume caustics in single-scattering media," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 2010, pp. 109–117.

[7] G. Liktor and C. Dachsbacher, "Real-time volumetric caustics with projected light beams," in *Proceedings of the 5th Hungarian Conference on Computer Graphics and Geometry*, 2010.

[8] ——, "Real-time volume caustics with adaptive beam tracing," in *Proceedings of the 2011 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 2011, pp. 1050–1053.

[9] C. Wyman and S. Ramsey, "Interactive volumetric shadows in participating media with single-scattering," in *IEEE Symposium on Interactive Ray Tracing*, 2008, pp. 87–92.

[10] H. W. Jensen, *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[11] T. J. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan, "Photon mapping on programmable graphics hardware," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 2003, pp. 41–50.

[12] L. Szirmay-Kalos, B. Aszdi, I. Laznyi, and M. Premecz, "Approximate ray-tracing on the gpu with distance impostors," *Computer Graphics Forum*, vol. 24, no. 3, pp. 695–704, 2005.

[13] C. Yao, B. Wang, B. Chan, J. Yong, and J.-C. Paul, "Multi-image based photon tracing for interactive global illumination of dynamic scenes," *Computer Graphics Forum*, vol. 29, no. 4, pp. 1315–1324, 2010.

[14] M. A. Shah, J. Konttinen, and S. Pattanaik, "Caustics mapping: An image-space technique for real-time caustics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 272–280, 2007.

[15] C. Wyman and G. Nichols, "Adaptive caustic maps using deferred shading," *Computer Graphics Forum*, vol. 28, no. 2, pp. 309–318, 2009.

[16] X. Yu, F. Li, and J. Yu, "Image-space caustics and curvatures," in *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pp. 181–188.

[17] I. Ihrke, G. Ziegler, A. Tevs, C. Theobalt, M. Magnor, and H.-P. Seidel, "Eikonal rendering: efficient light transport in refractive objects," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 59:1–59:9, July 2007.

[18] X. Sun, K. Zhou, E. Stollnitz, J. Shi, and B. Guo, "Interactive relighting of dynamic refractive objects," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 35:1–35:9, 2008.

[19] J. Kruger, K. Burger, and R. Westermann, "Interactive screen-space accurate photon tracing on gpus," in *EGSR*, 2006, pp. 319–329.

[20] X. Sun, K. Zhou, S. Lin, and B. Guo, "Line space gathering for single scattering in large scenes," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 54:1–54:8, July 2010.

[21] A. Boudet, P. Pitot, D. Pratmarty, and M. Paulin, "Photon splatting for participating media," in *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, 2005, pp. 197–204.

[22] C. Wyman, "An approximate image-space approach for interactive refraction," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1050–1053, July 2005.

[23] W. Hu and K. Qin, "Interactive approximate rendering of reflections, refractions and caustics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 46–57, 2007.

[24] B. Sheng, H. Sun, B. Liu, and E. Wu, "Gpu-based refraction and caustics rendering on depth textures," in *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, ser. VRCAI '09, 2009, pp. 139–144.

[25] T. Umenhoffer, G. Patow, and L. Szirmay-Kalos, "Robust multiple specular reflections and refractions," in *GPU Gems 3*, 1st ed., H. Nguyen, Ed. Addison-Wesley Professional, 2007.

[26] C. Everitt, "Interactive order-independent transparency," 2001. [Online]. Available: http://developer.nvidia.com/