Real-Time Caustics in Dynamic Scenes with Multiple Directional Lights

Budianto Tandianus, Henry Johan, Hock Soon Seah

Real-Time Caustics in Dynamic Scenes with Multiple Directional Lights

Budianto Tandianus, Henry Johan, and Hock Soon Seah

School of Computer Engineering, Nanyang Technological University,
N4 Nanyang Avenue, Singapore 639798
{budi0010,henryjohan,ashsseah}@ntu.edu.sg

Abstract. We present a real-time GPU caustics rendering technique in dynamic scenes under multiple directional lights taking into account light occlusion. Our technique renders caustics cast on receiver objects as well as volumetric caustics. We precompute caustic patterns of caustic objects for several directional lights and store them in caustic images. During the rendering, we interpolate the precomputed caustic patterns based on a given light direction. One of the applications of our technique is to render approximate caustics under environment illumination. To achieve this, we propose an environment cube map segmentation technique which divides cube maps into several light regions with each region is represented using one directional light.

Key words: Caustics, Real-Time Rendering, Environment Illumination, GPU

1 Introduction

Real-time photo-realistic rendering is a major goal in computer graphics and entertainment as it can help the audience to experience or even immerse into the virtual world as if it is the real world. However, it is computationally expensive to generate photo-realistic images. One of the important effects in photo-realistic rendering is caustics which are produced by reflective and/or refractive objects (**caustic objects**).

In this paper, we present a real-time caustics and volumetric caustics rendering technique under multiple directional lights taking into account light occlusion. Figure **??** shows some examples of our caustics rendering results. Our rendering technique can also be used to render approximate caustics under environment illumination. We achieve this by approximating the environment illumination using a set of directional lights computed using our proposed environment cube map segmentation technique. We precompute the reflective and/or the refractive caustic patterns at the surrounding of caustic objects based on a set of directional lights. We use the precomputed caustic patterns in the rendering pass in order to efficiently compute the caustic intensities at arbitrary locations. Our proposed technique has some differences with the technique presented by Wyman et al. [?] and we discuss these in Sections 2 and 3.

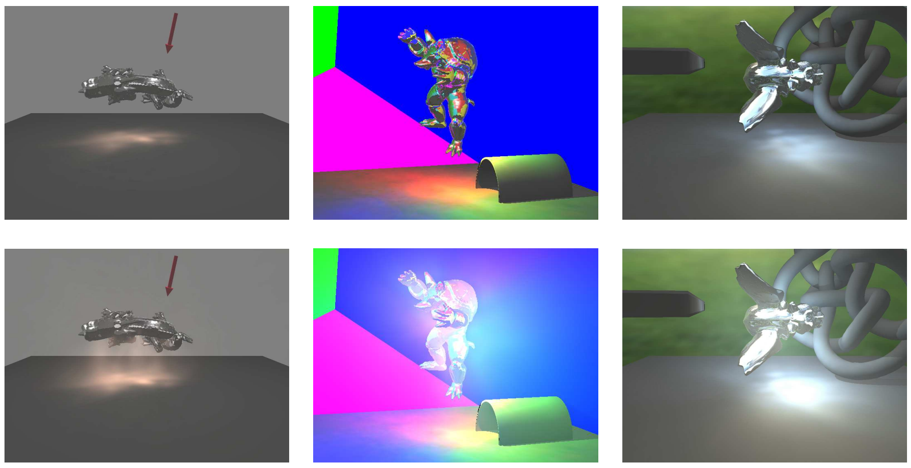


Fig. 1. Caustics rendering using our proposed technique. First column shows caustics under one directional light source (light direction indicated by the arrow). Second and third columns show caustics under environment illumination. The first row shows only the cast caustics, and the second row shows cast caustics and volumetric caustics.

2 Related Work

In photon mapping [?], photons (packets of light energy) are shot to the scene and stored in a photon map. The photon map is used during rendering by gathering the photons around a visible point in order to estimate the caustic intensity. However, it is computationally expensive and it needs to be recomputed if there are any changes in the scene. To accelerate the photon mapping, Günther et al. [?] use a computer cluster and they are able to achieve interactive rates whilst Purcell et al. [?] use a GPU and they are able to reduce the rendering time into few seconds. Zhou et al. [?] propose a real-time GPU-based kd-tree generation technique which greatly aids the photon mapping process. However, the overall rendering speed is mostly below 10 frames per second.

Another approach is the image-based caustics rendering proposed by Shah and Koninen [?], Sun et al. [?], Wyman and Davis [?]. In this approach, they shoot photons through each pixel by rendering the scene from the light source. In many cases, these techniques only support one light source to achieve interactive or real-time performance, do not support total internal reflection, and cannot generate volumetric caustics.

Wyman et al. [?] precompute the local caustics of a caustic object on uniform grids or concentric spheres (with constant radii differences between the consecutive spheres) enclosing the caustic object. They use a CPU cluster for both precomputation and rendering (their rendering speed is below 20 frames per second). In the rendering, the scene is illuminated by point or directional light sources, not environment illumination. On the other hand, our technique renders real-time caustics under environment illumination using a GPU.

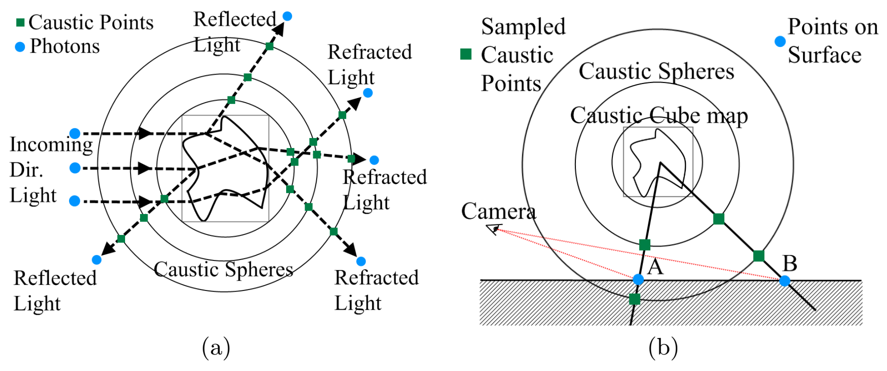


Fig. 2. (a) shows the precomputation of caustic patterns. The incoming directional light (implemented as photons) is reflected and refracted by the caustic object. (b) shows how we compute the caustic intensities on the visible points by interpolating or extrapolating the sampled points of nearby caustic spheres.

3 Precomputing Caustic Patterns

The purpose of the precomputation is to record the caustic intensities at the surrounding of caustic objects. We precompute the reflective and refractive caustics of a caustic object using photon mapping [?] on a set of concentric spheres (**caustic spheres**) based on a set of directional lights as illustrated in Figure **??**. Unlike Wyman et al. [?] who linearly change the radii of the spheres, we use a quadratic function to determine the radii of the caustic spheres since caustics weaken quadratically because of light attenuation. By using a quadratic function, we densely sample near to the caustic object and sparsely sample far from the caustic object. As a result, we can reduce the number of caustic spheres while maintaining the visual quality. Specifically, we use the following equation to compute the radii.

$$r_i = r_{\min} + (r_{\max} - r_{\min})((i-1)/(s-1))^2, \tag{1}$$

where r_i is the radius of the i-th caustic sphere, $r_{\min}$ is the minimum radius, $r_{\max}$ is the maximum radius, and s is the number of caustic spheres. We set $r_{\min}$ to be slightly greater than the distance from the center of the caustic object to the nearest surface of the caustic object. Assuming the maximum dimension of the object's bounding box d is max{width, height, depth}, we set $r_{\max}$ to $4d$ based on our experiments in which caustic patterns were barely noticeable on the caustic spheres with the radii of $4d$ and more.

4 Rendering Caustics under Directional Lights

Single Directional Light Given an arbitrary light direction L, we choose the four precomputed light directions nearest to L, rotate the caustic spheres of those light directions to align their directions with L (similar to Wyman et al. [?] who use three nearest light directions) and blend them using bilinear interpolation. Afterward, we use the computed caustic spheres of L to determine the caustic intensity at points in the scene. The caustic intensity at a point is trilinearly interpolated (point A in Figure **??**) or extrapolated (point B in Figure **??**) using the intensities at the eight nearest samples in the caustic spheres of L.

Caustics cast on receiver objects To render these caustics, we first compute a **caustic cube map** (Figure **??**) which stores the information of receiver points (the coordinates and depth value of the points) as well as their caustic intensities. Similar to shadow mapping, we compute those values by rendering the surrounding scene from the center of the caustic object and the caustics in the caustic cube map are then projected to the scene.

Volumetric caustics In the presence of participating media (assumed to be homogeneous), we generate volumetric caustics by casting a ray from every pixel on the screen to the scene [?] and we integrate the caustic intensity at each sample point on the ray.

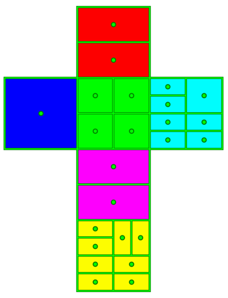
Multiple Directional Lights To generate caustics under D numbers of directional lights, we apply the algorithm for one directional light to each light and accumulate the caustic patterns on the caustic cube map.

5 Rendering Approximate Caustics under Environment Illumination

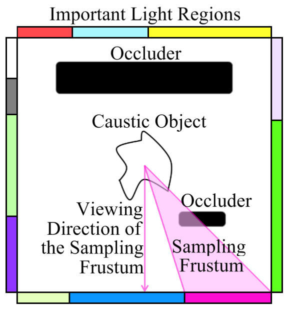
One of the applications of our caustics rendering technique is to approximate caustics under environment illumination. In this case, we represent the environment illumination as an environment cube map. Since integrating the radiance from all pixels in the cube map during rendering is impractical, we segment the environment cube map into several important light regions and represent each of them with a directional light. Thus, rendering caustics under environment illumination is similar to the rendering under multiple directional lights (Section **??**), that is integrating caustic patterns from all directional lights.

Environment Map Segmentation Debevec [?] recursively segments the environment map (latitude-longitude format) into two regions having almost equal total radiance until a number of iterations. In our technique, to account for light occlusion, we need to represent the environment illumination as a cube map. Therefore, we cannot directly use Debevec's segmentation algorithm [?].

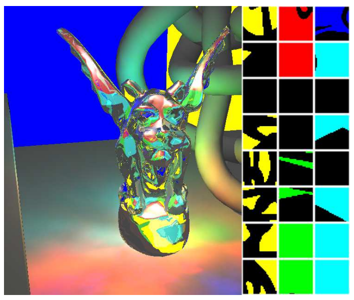
In our proposed environment cube map segmentation algorithm, in each segmentation iteration, we choose the region with the most radiance and segment it into two new regions having the same total radiance (similar idea as Debevec's segmentation [?]) until we obtain D light regions. We start the segmentation from six regions corresponding to each face of the cube map. For each light region, we store the weighted center (with the weight is the radiance of each pixel) of the region as the important light direction along with the region boundary information. Figure **??** shows the results of our environment cube map segmentation algorithm.



(a)



(b)



(c)

Fig. 3. (a) The result of our environment cube map segmentation with the lines are the region boundaries and the dots are the light directions representing the regions. (b) shows how we sample the important light regions for their radiance contribution by rendering them taking into account occluders (rendered as black color). (c) shows a scene configuration using the environment map in (a) with the sampling results shown on the right side. Note the occluders are rendered as black objects.

Directional Light Radiance Sampling As the directional lights are derived from an environment cube map, the radiance of each directional light is the total radiance of all unoccluded pixels in the important light region (corresponding to the directional light). To sample the regions by rendering, we set up the sampling frustum by using the boundary information of the important light regions (obtained from our environment cube map segmentation algorithm). As we sample the regions by rendering, we render the surrounding objects as black color in order to take into account light occlusion. Figure **??** illustrates how we sample the important light and Figure **??** shows an example of environment map sampling.

6 GPU Implementation of Caustics Rendering

We present the GPU implementation of our caustics rendering under D directional lights. Our implementation uses OpenGL and Cg as the rendering APIs. We use the image-based method proposed by Wyman et al. [**?**] to render the refraction effect of the caustic object.

Storing Caustic Spheres We store the precomputed caustic spheres (in latitude-longitude format) of each light direction in a 3D texture format so that we can directly use the trilinear interpolation provided internally by the graphics API when we sample the caustic spheres. We can store the caustic spheres either in **multiple 3D textures** (each 3D texture stores the caustic spheres of one light direction) or a **single 3D texture** (which tiles the caustic spheres of all light directions into one 3D texture). By using multiple 3D textures, we need to do multiple rendering passes (D passes) in order to accumulate the caustic patterns from all light directions. On the other hand, by storing in a single 3D texture we are able to accumulate the caustic patterns from all D light directions either in single pass or multiple passes.

Computing Directional Light Radiance The radiance for each light direction is computed by rendering the segmented regions of the environment map from the center of the caustic object. We render the segmented regions to a texture array whose resolution is $32 \times 32 \times D$, with each slice in the texture array corresponds to one segmented region. Afterward, we compute the total radiance by multiplying the number of pixels in that light region with the average radiance (computed by sampling the topmost level of the mipmap).

Caustics Sampling There are two ways for computing the caustic intensity from all lights. First, for each light we sample the four nearest precomputed caustic spheres directly (**direct sampling**) as explained in Section **??**. However, this becomes a bottleneck in volumetric caustics rendering since we need to do this process for every sampling point on the cast ray. Second, we compute and accumulate the caustic patterns of all lights beforehand (**compiled**) into a compiled 3D texture and then we just sample the compiled 3D texture.

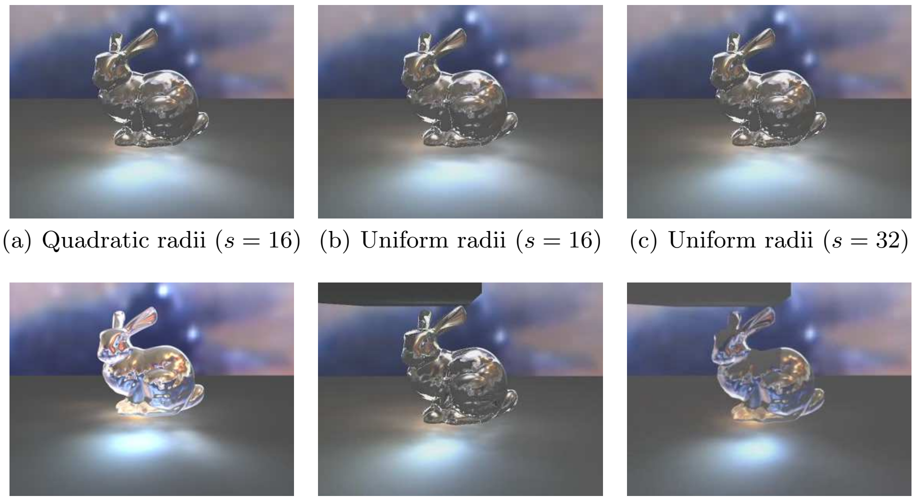
Rendering Passes The accumulation of the caustics from all light directions can be computed either in **multiple passes** or **single passes**. In **multiple passes**, the iteration is in CPU with each iteration corresponds to one directional light and we may store the precomputed caustic patterns in the GPU either in multiple 3D textures or in a single 3D texture. For **single pass**, the iteration through all D directional lights is performed in GPU and the information of all directional lights are stored in Texture Buffer Object. For this technique, we can store the precomputed caustic spheres in GPU only in a single 3D texture.

7 Results

We performed the experiments on a PC with an Intel Core i7 2.67 GHz and an Nvidia GTX 285. The image size of our real-time rendering results are 1024×768 pixels.

Rendering Results Figure **??** shows our rendering results under one directional light ($D = 1$ and $s = 16$, with D is the number of directional lights and s is the number of caustic spheres) and environment illumination ($D = 24$ and $s = 16$). From our experiments, the visual differences of the caustics between $D = \{32, 48\}$ and $D = 24$ were hardly noticeable. Moreover, the visual differences of $s = 16$ and $s = 32$ were also not very apparent. Therefore, we suggest the rendering with $D = 24$ and $s = 16$.

Sampling Comparison Figure **??** shows the comparison of caustics rendering using our quadratic radii caustic spheres and uniform radii caustic spheres proposed by Wyman et al. [?]. As seen in Figure **??**, the quadratic radii using fewer caustic spheres achieves similar visual results as the uniform radii which uses more caustic spheres. With few caustic spheres ($s = 8, 16$) circular banding artifact is visible near the caustic object in the rendering results using uniform radii caustic spheres due to the insufficient number of caustic spheres near the caustic object.



(a) Quadratic radii ($s = 16$) (b) Uniform radii ($s = 16$) (c) Uniform radii ($s = 32$)

(d) mental ray (without occlusion) (e) Occlusion - Quadratic radii ($s = 16$) (f) mental ray (with occlusion)

Fig. 4. Comparisons of the rendering results. (a), (b), (c), and (e) were rendered using 24 directional lights.

Comparison with mental ray We also compare our rendering results with the results rendered using mental ray (which took about four minutes to render each frame). Figures **??** and **??** show the comparison without light occlusion and Figures **??** and **??** show the comparison with light occlusion. Note the similarity between our results and mental ray results such as the caustics in front and behind the bunny (for the example without occlusion) and the vanished parts of the caustic patterns (for the example with occlusion). **Performance Comparison** We performed experiments using combinations of all possible options of the rendering techniques described in Section 6 and combinations of various numbers of directional lights $D = \{1, 16, 24, 32, 48\}$ and caustic spheres $s = \{8, 16, 32\}$ to determine the best rendering performance (in average frames per second, fps). In general, we achieved the best rendering performance under environment illumination for $D = 24$ and $s = 16$ (29.95 fps for cast caustics rendering and 14.57 fps for cast caustics and volumetric caustics rendering) by using the combination of multiple passes rendering, multiple 3D textures storage for the caustic spheres, and the compiled technique.

8 Conclusions and Future Work

We have presented a technique for real-time rendering of caustics and volumetric caustics in dynamic scenes under multiple directional lights. Our technique can be applied to render approximate caustics under environment illumination taking into account light occlusion from the surrounding objects. The limitations of the proposed technique are as follows. Our technique requires a large amount of memory to store the caustic spheres. Thus, we are interested in finding a method to compress the caustic spheres. Since we perform precomputation, the caustic objects are not deformable.

Acknowledgments. This work has been supported by the National Research Foundation grant, which is administered by the Media Development Authority Interactive Digital Media Programme Office, MDA (IDMPO). We would like to express our gratitude to Paul Debevec for the environment cube maps (Campus, Kitchen, and Grace Cathedral) and Stanford Computer Graphics Laboratory for the 3D models (bunny, dragon, armadillo).

References

1. Debevec, P.: A median cut algorithm for light probe sampling. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Posters, p. 66. ACM, New York, NY, USA (2005)
2. Günther, J., Wald, I., Slusallek, P.: Realtime caustics using distributed photon mapping. In: A. Keller, H.W. Jensen (eds.) Rendering Techniques 2004 : Eurographics Symposium on Rendering, pp. 111-121. Eurographics Association, Eurographics, Norkoeping, Sweden (2004)
3. Hadwiger, M., Ljung, P., Salama, C.R., Ropinski, T.: Advanced illumination techniques for GPU volume raycasting. In: SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 Courses, pp. 1-166. ACM, New York, NY, USA (2008)
4. Jensen, H.W.: Global illumination using photon maps. In: Proceedings of the Eurographics Workshop on Rendering Techniques '96, pp. 21-30. Springer-Verlag, London, UK (1996)
5. Purcell, T.J., Donner, C., Cammarano, M., Jensen, H.W., Hanrahan, P.: Photon mapping on programmable graphics hardware. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Courses, p. 258. ACM, New York, NY, USA (2005)
6. Shah, M.A., Konttinen, J.: Caustics mapping: An imagespace technique for real-time caustics. IEEE Transactions on Visualization and Computer Graphics 13(2), pp. 272-280 (2007)
7. Sun, X., Zhou, K., Stollnitz, E., Shi, J., Guo, B.: Interactive relighting of dynamic refractive objects. In: SIGGRAPH '08: ACM SIGGRAPH 2008 Papers, pp. 1-9. ACM, New York, NY, USA (2008)
8. Wyman, C., Hansen, C., Shirley, P.: Interactive caustics using local precomputed irradiance. In: PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference, pp. 143-151. IEEE Computer Society, Washington, DC, USA (2004)
9. Wyman, C.: Interactive image-space refraction of nearby geometry. In: GRAPHITE '05: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, pp. 205-211. ACM, New York, NY, USA (2005)
10. Wyman, C., Davis, S.: Interactive image-space techniques for approximating caustics. In: I3D '06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, pp. 153-160. ACM, New York, NY, USA (2006)
11. Zhou, K., Hou, Q., Wang, R., Guo, B.: Real-time KD-tree construction on graphics hardware. ACM SIGGRAPH Asia 2008 papers. pp. 1-11 ACM, Singapore (2008).