AN INTERACTIVE METHOD FOR REFRACTIVE WATER CAUSTICS RENDERING USING COLOR AND DEPTH TEXTURES

Nuttachai Tipprasert
Department of Computer Engineering,Faculty of Engineering,Chulalongkorn University 254 Phyathai Road, Patumwan, Bangkok Thailand. 10330

Pizzanu Kanongchaiyos
Department of Computer Engineering,Faculty of Engineering,Chulalongkorn University 254 Phyathai Road, Patumwan, Bangkok Thailand. 10330

Keywords: global illumination, caustics, natural phenomena, color texture, depth texture, interactive rendering.

Abstract: Realistic rendering of underwater scenes is one of the most anticipated research topics in computer graphics. Caustics are the important component enhancing the realism of this kind of scenes. Unfortunately, rendering caustics is a time consuming task. As a result, most existing algorithms cannot handle this at interactive rate. In recent years, volumetric texture based rendering algorithms have been proposed. They can render the underwater scene with caustics in real-time. However, these algorithms require large amount of memory and are restricted to non-complex scene. In this paper we present a new interactive caustics rendering algorithm which require less memory usage. In our proposed method, we represent each object as a pair of color and depth texture. Color texture is used to store the object image viewed from viewing rays which refracted at water surface. We calculate the light intensity distribution on this image and store the result back to the color texture. The depth texture is used in the intensity calculation process to improve accuracy of the caustics patterns. Our experiment shows that proposed algorithm can handle complex underwater scene with caustics at interactive time rate. While using a pair of color and depth in stead of volumetric texture, we can reduce memory usage significantly.

1 INTRODUCTION

Research in the filed of realistic natural phenomena rendering is one of the most important subjects in computer graphics. Of all research topics, the realistic rendering of scene with water is one of the most anticipates research topics in this filed. To enhance the realism of this kind of scene, caustics rendering is one of the most important aspect that must be taken into account. But the rendering process of this phenomenon involves many path tracings and intersection tests. As a result, the rendering of a realistic water scene seems to be more suitable for off-line rendering rather than real-time rendering. However, there are many applications, such as video games and virtual realities which require realistic real-time rendering of such a scene. Therefore, the traditional rendering algorithm cannot be employed at these applications.

To reduce the computation cost, the volumetric texture based caustics rendering algorithm has been proposed (Iwasaki, 2003, Iwasaki, 2005). This technique use volumetric textures to represent the objects in the scene and perform the intersection test on these textures instead. Even though the algorithm can achieve interactive rendering capability, it requires a lot of memory. As a consequence, these techniques are limited to a simple scene that doesnot have many objects.

Due to this limitation, we introduce a new interactive method for rendering underwater scene with caustics as viewed from above water. Our technique requires less memory usage. In our proposed method, the objects are represented by one pair of color and depth texture .These textures are used in both caustics casting and refracted objects rendering processes to enhance the performance. Color texture is used to store the object image viewed from viewing rays which refracted at water surface. The depth texture is used to represent 3D position of each pixel in color texture. The algorithm is accelerated by performing intersection and computing intensity distribution on texture-space in stead of object-space. We are able to show that this technique can generate complex underwater scene with caustics at interactive time-rate.

The remainder of this paper is organized as follows: Next section, we will briefly discuss the related work. In section 3, a main concept of our rendering strategy will be presented. Section 4 our sample results are shown and then we gave the conclusion and future work in section 5.

2 RELATED WORK

In the past few decades, many algorithms have been developed to simulate global illumination effects such as caustics. (Arvo, 1986; Jensen, 1996; Guenther, 2004; Trendall, 2000; Wald, 2002; Wyman, 2004) Even though these techniques can render realistic caustics, they require long computational time or require special hardware setting. Recently, Shah et al. (Shah, 2005) presented real-time caustics rendering algorithm based on backward ray-tracing. In order to speed up the algorithm, they created position texture and used it to store 3D world coordinate of each object in the scene, then, perform the intersection tests in the image-space. The caustics pattern is rendered by using point primitive. Though, the main concept of their algorithm similar to our work, the whole idea has so many differences in details. Besides, their algorithm suffers from alias problem, just like any other image-space algorithms.

There are several methods that developed for underwater caustics rendering. Stam (Stam, 1996) simulated underwater caustics by generating caustics textures and mapping them onto objects in the scene, Crespo (Crespo, 2004) has proposed a method that was extended from this concept and implemented it on programmable graphics hardware, although these methods can simulate underwater caustics in real-time, the results are not visually correct due to the fact that they perform light intensity distribution calculation on flat surface. Watt (Watt, 1990) introduced underwater caustics rendering algorithm using backward beam-tracing, which was extended from the algorithm originally proposed by Heckbert (Heckbert, 1984). Rather than tracing individual light rays, the backward beam-tracing traces light beam that emerge from light source and then refracts them at each polygons of water mesh. The caustics patterns are generated by accumulated light intensity that each receiver polygons receives from each participated light beam. Though the beautiful images of underwater scene can be generated from this algorithm, the computation time is also extremely long. The main problem about beam-tracing based caustics rendering algorithm is the intersection test between light beam and diffuse receiver. Nishita and Nakamae (Nishita, 1994) solved this problem by subdividing light beam and using scan-line algorithm to determine intersection point. Their algorithm was then improved by Iwasaki et al. (Iwasaki, 2002). In the following works, Iwasaki et al. applied volume rendering technique to handle the case where the observers are above the water (Iwasaki, 2003). Their proposed method creates slice image of each receiver object in which the caustics pattern that cast on these objects can be depicted by performing the intersection test of light beam on these images. They continue working on this method by presenting the extended algorithm for casting caustics from arbitrary refractive medium (Iwasaki, 2005). By performing intersection test on the collection of slice images instead of object mesh, the computation time is greatly reduced. However, these algorithms require large amount of texture memories; as a consequence, they are not suitable for using with complex scene.

To address this problem, our algorithm replaces the usage of volumetric texture by using a pair of color and depth texture. Our algorithm can display refractive caustics due to water surface at interactive frame-rate and requires much less memory usage than the previously proposed beam-tracing based interactive caustics rendering techniques.

3 RENDERING ALGORITHM

In this section we give a description of our new algorithm. First we present the general idea of fast intersection test by using a combination of depth texture and reference planes (3.1). Next, we describe how to render underwater scene with caustics. In this step, the color texture then comes into play (3.2). Finally, the discussion on rendering refracted underwater scene is given (3.3).

3.1 General Idea of Fast Intersection Test Technique

In order to optimize the rendering speed of caustics rendering algorithm, the method for testing intersection between light beam and objects must be improved. Iwasaki et al. (Iwasaki, 2003) solved this problem by using volumetric textures. In their proposed method, the volumetric textures are created by projecting a part of objects images that lie between two adjacent virtual planes onto one of these planes. They referred to these virtual planes as "Sampling Plane". The sampling plane is used in the intersection test step to determine the intersection point of light beam and viewing ray at receiver geometry. Because the number of sampling plane for each objects are much less than the number of polygons, the iteration steps required for finding intersection point are extremely reduced. Nevertheless, this algorithm still has a memory usage problem because large amount of texture memory is required to store these images. We observed that if we line up these images in the correct order from back to front and look straight through them, these images will appear as one complete image. From this observation, we realize that only one color texture is sufficient to represent a diffuse object. However, the use of single 2D image to represents 3D object data cannot conserve volumetric property of the object. If we estimate intersection point by testing intersection on this image, the result may be undesirable. The ray may "early hit" the image plane or "lately miss" the actual intersection point (see figure 1). We solve the problem mention above via the use of depth texture and *Reference Planes*. In the proposed method, the depth texture is used to store 3D position of each pixel of object geometry and the reference planes is the plane which virtually slices along some given major axe of object.

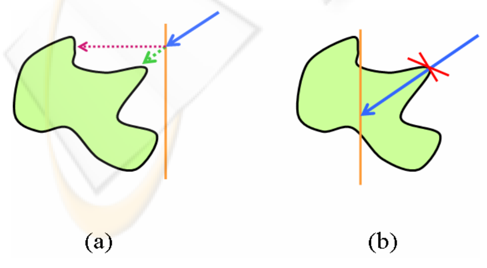

Figure 1: Incorrect intersection estimation. (a) The incoming ray hit image plane early. Figure (b) show the case where the ray is misses the actual intersection point.

In the intersection test step, we use these reference planes for indexing texture value. When a ray intersects with the reference plane, the intersection point (x, y, z) is then transform into texture space coordinate (x', y', z'). The x' - and y' - coordinate are used to index appropriate entry of the depth texture and the resultant value is compared against value of z'. If the differences of this two values are less than some specific threshold, this intersection point is then accepted (see figure 4). Normally, this acceptance threshold should be the distance between two reference planes. Otherwise, some undesirable result might occur.

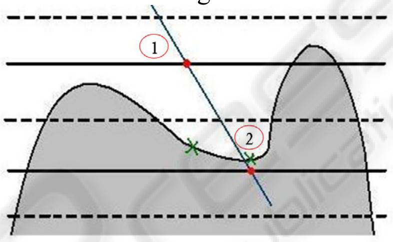

Figure 2: Diagram of our intersection test algorithm. The opaque line represent reference planes, dash line represent acceptance range of each reference plane and gray curve represent object surface. From this image, the intersection point that will be accepted is the second point.

From this proposed algorithm, the intersection test can be done at interactive time-rate and does not require extra memory usage as it did in the previous method. In the upcoming sections, we will show how this technique can be used in underwater caustics rendering process.

3.2 Rendering Caustics

The caustics pattern are formed by multiple refracted light rays converge to single point on diffuse object geometry. In our proposed algorithm, we emulate this behavior by representing water surface as triangular mesh. When the incident light rays intersect with each water triangle, they create refracted light beam called *illumination volume*. Next, we find intersection area of each illumination volume and object geometry and compute light intensity for each of them. The intensity of intersection area can be computed from radiant equation:

$$I_c = \Phi_t / (A_d * \cos(\theta_t)) \qquad (1)$$

where Φ_t is the total flux that arriving intersection area A_d and θ_t is the angle between the refracted light ray and the normal of intersection area. The value of Φ_t in equation (1) can be obtained by finding total flux Φ_i that pass through water triangle. When light travels through water, some of its energy are absorbed. Thus, the relationship between Φ_t and Φ_i can be written as:

$$\Phi_t = \Phi_i * exp(-K*d) \qquad (2)$$

where K is the absorption coefficient and d is the distant light travels through the water. Let I_i be incident light intensity. By substituting equation (1) with equation (2) and representing Φ_i in term of I_i, we get:

$$I_c = I_i * ((A_w * cos(\theta_i) / (A_d * cos(\theta_t)) * exp(-K*d) \qquad (3)$$

where A_w is an area of water triangle and θ_i is an incident light angle. By accumulating the intensity of each participated intersection area, caustics pattern can be depicted. Finally, the final color of each pixel on diffuse receiver is computed from this equation:

$$I_o = I_c * I_d + I_a \qquad (4)$$

where I_o is the final color of the object I_d is diffuse light intensity and I_a is ambient light intensity.

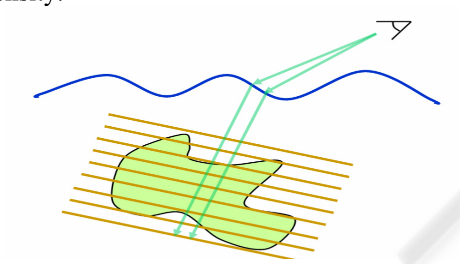

Figure 3: Reference plane alignment.

In order to cast caustics pattern onto receiver geometry by the method mentioned above, the intersection test problem must be addressed. We use the fast intersection test strategy described in the previous section to handle the case. To find the intersection area, we first create depth texture and reference planes. In the process of depth texture and reference planes creation, we first specify the bounding box of an object. This bounding box is also used to define orthogonal view-volume which will be used in the next step. We must align this bounding box by setting the front side of the box perpendicular to the refracted viewing ray (see figure 3). Then we create depth textures by rendering diffuse object using orthographic projection. We only store the position of each vertex and copy them to the depth texture. The reference planes are determined by slicing the bounding box along the refracted viewing ray. The number of sliced planes is specified by user.

After the depth texture and reference planes are created, we move to the next step; caustics rendering. In our proposed method, we first create diffuse image of the receiver by taking the refracted viewing ray into account as in the case of depth texture. We call this image a *Diffuse Map* and store it into color texture. Next, we find the intersection area between each illumination volume and each reference plane. We then draw intersection triangle of each area to a color texture called *Caustics Map* by using additive blending function. The color of each triangle vertex is determined by calculating intensity at the intersection point. After the intersection triangle is rasterized, we transform each of their pixels into the depth texture space. The transformed coordinate of this pixel will be used to index the value of depth texture. This indexed value is then compared with pixel's transformed z coordinate. If the differences of these two values are less than specific threshold, we accept this pixel; otherwise, we discard it. After we finish with caustics map creation, we then multiply it with diffuse map to obtain final result. This result is then stored back to the diffuse map. The overall processes of caustics rendering are visualized in the figure 4.

By using this proposed method the image of underwater caustics can be generated at interactive time-rate. In order to create complete underwater scene as viewed from the above water, there is one problem unsolved, that is, refracted underwater image rendering. Unfortunately, current real-time refracted image rendering method such as environment mapping cannot be applied to our algorithm. This limitation comes from the fact that our algorithm create object image by using orthographic projection. If we directly apply these images to water vertex as environment textures, the resulting image may be undesirable. Therefore, new techniques for handling refracted image of underwater scene created by our algorithm must be proposed. This technique will be described in details in the upcoming subsection.

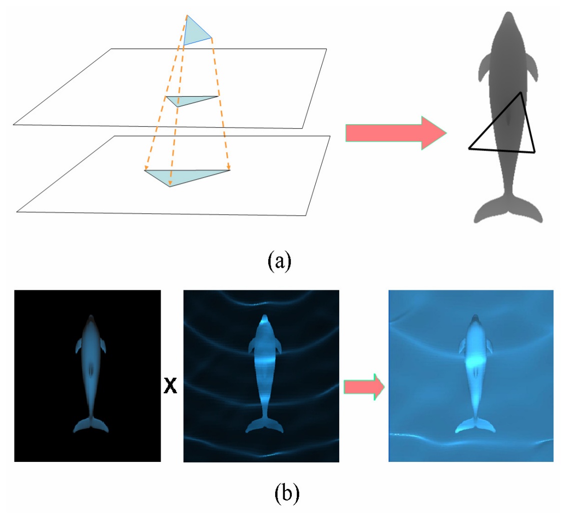

Figure 4: Caustics rendering process. In figure (a), the triangle represents intersection area for each reference plane. Only accepted fragments of each intersection triangle are drawn to the caustics map. In figure (b), the final image is computed by multiplying diffuse map and caustics map.

3.3 Rendering Underwater Image as Viewed from Above Water

In this section, we describe the method for rendering complete underwater scene with caustics as viewed from above water.

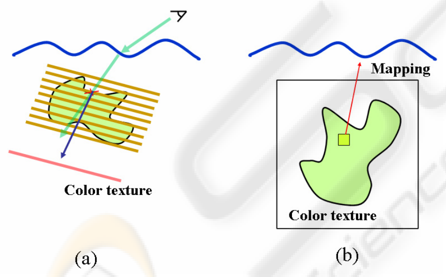

Figure 5: Refraction mapping technique.

In order to render the refracted image of under water objects, our method applies the concept of ray-tracing and texture mapping (see figure 5). The proposed method begins by calculating the viewing vectors from view point to each fragment of water triangle. Next, we generate refracted viewing ray at the water surface. After that, the viewing ray is traced and checked for the intersection point. The intersection test is performed by using our proposed reference plane and depth texture strategy. The final color of water fragment is specified by transforming the intersection point to texture space and obtaining color from diffuse map.

4 IMPLEMENTATION AND RESULTS

There are so many fragment operations involved in our algorithm; as a result, it cannot be applied to the fixed function operation on traditional graphics hardware. By taking an advantage of programmability on this day GPU, we can perform caustics rendering using our algorithm at interactive time-frame. We test our idea by implementing our proposed algorithm on 2.8 GHz Pentuim IV desktop with Geforce 6800 128 Mb GPU. The test programs are written in C++ using OpenGL API. Figure 6 show the image of underwater teapot and dolphin. All images are rendered at resolution of 512 x 512 pixels. Average rendering time of our program is about 2.2 fps depend on number of reference plane.

The sample images indicated that our algorithm can cast caustics on arbitrary objects. Two images on figure 6(b) are rendered by using different number of reference plane. Although the reference plane of these two images is 10 times differing, the differences on the resulting image are hardly noticeable. It can be concluded that, in some objects small, number of reference plane are sufficient to create realistic result.

All sample programs can run in interactive time-rate. Although in this version of our program, we have not yet shown dynamic scene, such a case would not be prone to any problem from our algorithm, because in this version both color and depth textures are recreated at each frame.

By comparing with previously proposed volumetric texture based method (Iwasaki, 2003; Iwasaki, 2005), the recreation of volumetric texture can cause a problem. And by using only single pair of color and depth texture to represent an object, the required memory usage from the previously proposed method can be reduced significantly. As a result, our algorithm is not restricted to a simple scene.

5 CONCLUSION AND FUTURE WORK

We have presented a new method for interactive rendering underwater scene with caustics. With our proposed method, the problem of complex scene in previous method (Iwasaki, 2003) has been solved.

The main limitation of our algorithm is it can only handle refractive caustics and underwater images. Thus, it cannot be used to generate complete water-side scene. In order to create realistic water-side scene, the reflective caustics due to water surface and reflective image of objects must be taken into consideration. The casting process of reflective caustics is somewhat different from refractive caustics, thus, this algorithm must be extended in order to handle such an effect in future work. Furthermore, we want to find the optimization method for this proposed algorithm. From our observation, the LOD method may be used to apply with our reference plane algorithm by using less slice plane at the distance object. The work on applying this concept is still in development.

REFERENCES

Arvo, J. (1986) Backwards Ray Tracing. *SIGGRAPH' 86 Course Note,* 12**,** 259-263.

Crespo, D. S., & Guardado, J. (2004) *Rendering water caustics*, Addison Wesley.

Guenther, J., Wald, I., & Slusallek, P. (2004) Realtime caustics using distributed photon mapping. *Eurographics Workshop on Rendering*

Heckbert, P. S., & Hanrahan, P. (1984) Beam tracing polygonal objects. *SIGGRAPH' 84.* ACM Press.

Heidrich, W., & Seidel, H. (1998) View-independent environment maps. *Graphics Hardware.*

Iwasaki, K., Dobashi, Y., & Nishita, T. (2002) An efficient method for rendering underwater optical effects using graphics hardware. *Computer Graphics Forum,* 21**,** 701-711.

Iwasaki, K., Dobashi, Y., & Nishita, T. (2003) A fast rendering method for refractive and reflective caustics due to water surfaces. *Computer Graphics Forum,* 22**,** 601-609.

Iwasaki, K., Yoshimoto, F., Dobashi, Y., & Nishita, T. (2005) A Method for Fast Rendering of Caustics from Refraction by Transparent Objects. *IEICE Transaction,E88-D: Special Issue on CyberWorlds,* 5**,** 904-911.

Jensen, H. W. (1996a) Global illumination using photon maps. *Rendering Techniques' 96.* Springer-Verlag/Wien.

Jensen, H. W. (1996b) Rendering caustics on non-lambertian surfaces. *Graphics Interface' 96*

Nishita, T., & Nakamae, E. (1994) Method of displaying optical effects within water using accumulation-buffer. *SIGGRAPH' 94.* ACM Press.

Shah, M. A., & Pattanaik, S. (2005) Caustics Mapping: An Image-space Technique for Real-time Caustics.

Stam, J. (1996) Random caustics: natural, textures and wave theory revisited. *SIGGPRAPH' 96.* ACM Press.

Trendall, C., & Stewart, A.J. (2000) General calculation using graphics hardware, with application to interactive caustics. *Eurographics Workshop on Rendering.*

Wald, I., Kollig, T., Benthin, C., Keller, A., & Slusallek P. (2002) Interactive Global illumination using fast ray tracing. *Eurographics Workshop on Rendering*

Watt, M. (1990) Light-water interaction using backward beam tracing. *SIGGRAPH' 90.* ACM Press.

Wyman, C. (2005) An approximate image-space approach for interactive refraction. *SIGGRAPH 2005.* ACM Press.

Wyman, C., Hansen, C. D., & Shirley, P. (2004) Interactive caustics using local precomputed irradiance. *Pacific Conference on Computer Graphics and Applications.*

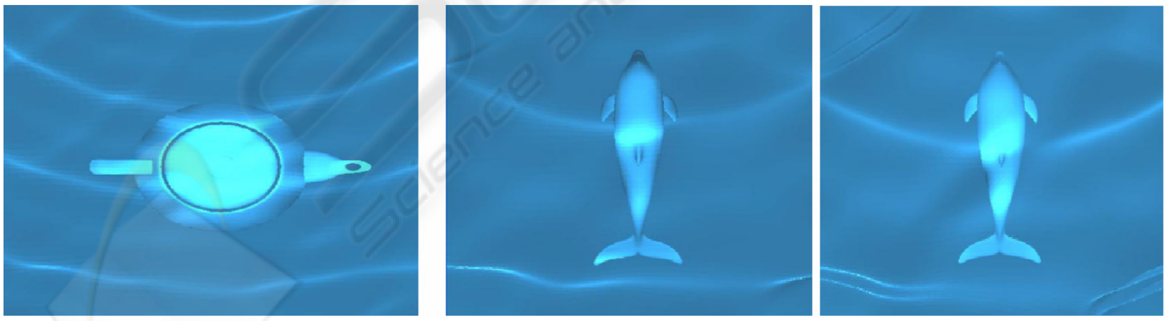

Figure 6: Results from our algorithm. Figure (a) show under water teapot. Figure (b) compare two dolphin images which rendered by using different number of reference plane. Image on the left was rendered at average 2.2 fps with 50 reference planes. Image on the right was rendered at average 10.7 fps with 5 reference planes.