Vol.2019-CG-174 No.3
2019/6/27

Antialising for Rendering Reflection on Water Surface

Podee Namo[1,a) Nelson Max[2,b)] Kei Iwasaki[3,c)] Yoshinori Dobashi[1,d)]

概要：A reflection of a bright light source on a dynamic surface such as a water surface can be difficult to render in high-quality in real-time due to reflection aliasing and flickering. In this paper, we propose a solution to this problem by approximating the reflection distribution of any aliasing prone water surface as a Gaussian distribution. Then we analytically integrate the reflection contribution throughout the rendering interval time. Our method is able to render a reflection of a spherical light source on highly dynamic waves with less aliasing and unnatural flickering in real-time.

1. Introduction

The ocean surface is highly dynamic. It moves rapidly and thus its shading changes rapidly as well. Usually, this doesn't pose any problems if the shading is smooth. However, for a surface that has a strong highlight or bright reflection moving rapidly, it causes an inaccurate and unnatural flickering. In the traditional rendering algorithms, each frame is rendered independently at a discrete time, resulting in serious temporal aliasing artifacts. Particularly, for a wavy water surface, reflection vectors may not hit the light source even though they actually hit for part of the frame time. Removing such aliasing in real-time is an active research area and many methods have been proposed [1]. They can improve the fidelity and efficiency of the rendering method. However, their focus is on spatial anti-aliasing and most of them do not address the temporal aliasing problem, particularly the one observed in rendering a reflected image of a light source on the water surface.

In this paper, we present a method that can remove the spatial and temporal aliasing simultaneously. The basic idea is to compute the intersection of the light source with a plane formed by two reflection vectors for neighboring frames. This provides us with a fraction of time when the light source is visible on the water surface. We combine this idea with a traditional spatial anti-aliasing method.

[1] Hokkaido University
Kita-ku, Kita 14, Nishi 9, 060-0814, Sapporo, Japan
[2] Univeristy of California at Davis
1 Shields Ave, Davis, CA 95616, USA
[3] Wakayama University
Sakaedani 930, Wakayama, Japan
[a)] namo@ime.ist.hokudai.ac.jp
[b)] max@cs.ucdavis.edu
[c)] iwasaki@sys.wakayama-u.ac.jp
[d)] doba@ime.ist.hokudai.ac.jp

2. Related Work

There are three main groups of works related to our research: reflection rendering, screen space anti-aliasing, and geometric analytic anti-aliasing.

Reflection rendering: One of the most popular methods for rendering a reflection in real-time is image-based rendering. A method proposed in [2] renders realistic reflection images at an interactive speed by precomputing radiance maps that store the product of an environment map and a BRDF to efficiently compute the outgoing intensity of light for an arbitrary surface orientation and viewing direction. Since this method cannot change either surface materials or lighting conditions at the run time, McAuley et al. [3] introduce a split-sum approach that approximately computes the product of the environment map and the BRDF efficiently. A more recent work achieves the real-time calculation of the shading due to a polygonal light source by linear transformation of cosine lobes [4] and a spherical light source by spherical distribution transformation that preserves spherical cap called a pivot [5]. However, none of these methods pay any attention to the temporal aliasing.

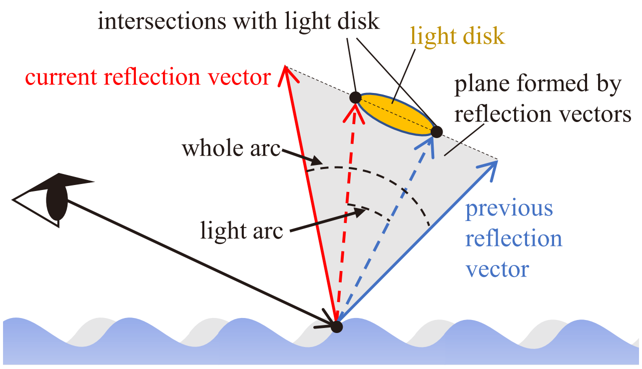

図 1 Basic idea of our method.

Screen space: Anti-aliasing is an active research area due to its importance in the fidelity of real-time rendering. Fast approximate anti-aliasing (FXAA) [6] is one of the most popular anti-aliasing methods. It simply detects jagged edges in an image and smoothes them. It is fast but it blurs some details in the final image. Temporal anti-aliasing has also been studied for a long time [7], [8]. These methods determine pixels of moving objects and then perform filtering on them.

Analytic anti-aliasing: Our method is categorized in this group. The methods in this group analyze the cause of the aliasing and find a mathematical solution for each target aliasing situation. A clamping method [9] removes aliasing on a textured surface by using the Nyquist theorem, which we will use in our method. The method proposed in [10] analyzes the movement of a polygon in the image space and generates a space-time representation of the object for spatial and temporal anti-aliasing. EWA volume splatting [11] is an anti-aliasing method for volume rendering. It uses the elliptical Gaussian function as a reconstruction kernel of the volume to avoid aliasing. We are inspired by this work and use the Gaussian kernel's affine mapping. Our method also builds on a method called LEAN mapping [12] that approximates the distribution of surface normal vectors with a Gaussian function to eliminate aliasing that comes from using a bump map. However, these methods do not take into account the temporal aliasing, which we address in this paper.

3. Our Approach

Our goal is to compute the contribution of a spherical light source over a period of time between rendering frames. We assume that the water wave is represented as a sum of sine waves with different frequencies and directions. For each pixel, our method first decomposes the water wave into two frequency bands: low and high-frequency bands, namely non-aliasing waves and aliasing waves, respectively. The high-frequency components, or aliasing waves, cause spatial aliasing due to under-sampling. We develop two methods, non-aliasing wave rendering and aliasing wave rendering, for these two frequency bands, respectively. The fundamental of both methods is the same but the spherical light source is blurred to account for the effects of the BRDF caused by the aliasing waves.

3.1 Aliasing detection

We use the clamping anti-aliasing method [9] to decompose the water waves into aliasing and non-aliasing waves. According to sampling theory, to avoid aliasing the sampling frequency must be higher than the twice the highest frequency of the water wave. In our case, we calculate the projected wavelength of each sine wave on each screen pixel. Each of the sine waves is then classified into either the non-aliasing and or the aliasing waves according to the Nyquist frequency, which is $\frac{1}{2\sqrt{2}}$ pixels. Any projected wave that has lower frequency is a non-aliasing wave and vice versa for aliasing wave. We use soft classification, with a smooth amplitude transition starting before the Nyquist limit, to avoid arcs in the image of sudden appearance changes.

3.2 Non-aliasing wave rendering

For the non-aliasing waves, we sample a single point on the water surface corresponding to the pixel center and compute the contribution of reflected light over the time interval between the current and the previous frames. We assume that the light source is far distant from the sample point and is approximated by a disk facing toward the sample point. Two reflection vectors are computed by using the normal vectors at the previous and the current frames. We then calculate intersection points between the light disk and a plane formed by the two reflection vectors, as shown in Fig. 1. A reflected viewing ray at the sample point hits the light disk when it lies between the directions to the two intersection points. Thus, the fractional contribution of the light source between the frames is obtained by the ratio of the angle of the light arc to the angle of the whole arc (see Fig. 1).

3.3 Aliasing wave rendering

For aliasing waves, a single sample point per pixel is not sufficient. We have to compute the average intensity of the reflected light over the pixel area taking into account the distribution of the surface normals. A straightforward solution is to generate multiple rays for each pixel, which significantly increases the computation time. Instead, we borrow the idea of the LEAN mapping technique [12] for efficient computation.

We first calculate the covariance matrix of the normal distribution function (NDF). The covariance matrix for the sum of the sine waves is obtained by accumulating the covariance matrix for each of the waves, which can be calculated analytically by integrating the normal direction of the whole sine wave.

Then we transform the NDF into the reflection space to obtain the reflection distribution function (RDF). The RDF represents the distribution of the reflection directions. We are inspired by [11], which transforms a Gaussian kernel from camera space to ray space by using a local affine approximation. We represent both NDF and RDF as elliptical Gaussians and use the local affine transformation to approximate the RDF. The convolution of the RDF and a light disk is the light contribution of each normal direction. To avoid temporal aliasing, we need to integrate a light contribution of a moving normal direction over time, and we assume that the normal direction to the non-aliasing waves is moving linearly over time. Then the temporal contribution of the convolution is computed by a line integral along the moving direction of the normal vector. However, it is costly to compute this in real-time.

To be efficient, we precompute the line integration over the convolution of RDF and a light disk and save it into a texture with just three parameters: the perpendicular distance of the extended line segment to the center of the light disc, the distance along the line from the foot of this perpendicular to an endpoint of the segment, and the variance of the RDF. We approximate the elliptical Gaussian RDF as a circular Gaussian to reduce its dimension, which now needs only one variance parameter. Then the configuration becomes circularly symmetric, so only two parameters are needed for the line integral over the blurred disc.

4. Results and Conclusion

Fig. 2 shows that our method can significantly reduce aliasing artifacts. Please see the accompanying video for the animated version of this example. The reference image is created by generating an image with 64 times higher resolution and then downsampling it. For the temporal anti-aliasing, we generate 8 images between the neighboring frames and compute their average. The rendering time for our method and the reference images are 27 and 17214 ms respectively. These are measured on a laptop with Intel Core i7 @ 2.50Ghz, Memory 16 GB, and NVIDIA GeForce GTX 860M.

Our method reduces aliasing and increases the fluidity of wave reflection animation in real-time by using temporal and spatial anti-aliasing methods. It also deals with the changing position of a light source and works for any height/normal field, if its normal distribution is known for its aliased wave. However, by approximating the RDF as a circular Gaussian, our method loses accuracy for distant waves, which have more directionality. We are planning to address this issue by approximating the elliptical Gaussian with a set of circular Gaussians.

謝辞 This work was supported by JSPS KAKENHI Grant Number JP15H05924.

参考文献

[1] Jimenez, J., Gutierrez, D., Yang, J., Reshetov, A., Demoreuille, P., Berghoff, T., Perthuis, C., Yu, H., McGuire, M., Lottes, T., Malan, H., Persson, E., Andreev, D. and Sousa, T.: Filtering Approaches for Real-time Anti-aliasing, *ACM SIGGRAPH 2011 Courses*, SIGGRAPH '11, New York, NY, USA, ACM, pp. 6:1–6:329 (online), DOI: 10.1145/2037636.2037642 (2011).

[2] Cabral, B., Olano, M. and Nemec, P.: Reflection Space Image Based Rendering, *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co., pp. 165–170 (online), DOI: 10.1145/311535.311553 (1999).

[3] McAuley, S., Hill, S., Martinez, A., Villemin, R., Pettineo, M., Lazarov, D., Neubelt, D., Karis, B., Hery, C., Hoffman, N. and Zap Andersson, H.: Physically Based Shading in Theory and Practice, *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, New York, NY, USA, ACM, pp. 22:1–22:8 (online), DOI: 10.1145/2504435.2504457 (2013).

[4] Heitz, E., Dupuy, J., Hill, S. and Neubelt, D.: Real-time Polygonal-light Shading with Linearly Transformed Cosines, *ACM Trans. Graph.*, Vol. 35, No. 4, pp. 41:1–41:8 (online), DOI: 10.1145/2897824.2925895 (2016).

[5] Dupuy, J., Heitz, E. and Belcour, L.: A Spherical Cap Preserving Parameterization for Spherical Distributions, *ACM Trans. Graph.*, Vol. 36, No. 4, pp. 139:1–139:12 (online), DOI: 10.1145/3072959.3073694 (2017).

[6] Lottes, T.: FXAA, `http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf` (2009).

[7] Korein, J. and Badler, N.: Temporal Anti-aliasing in Computer Generated Animation, *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '83, New York, NY, USA, ACM, pp. 377–388 (online), DOI: 10.1145/800059.801168 (1983).

[8] Shinya, M.: Spatial Anti-aliasing for Animation Sequences with Spatio-temporal Filtering, *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, New

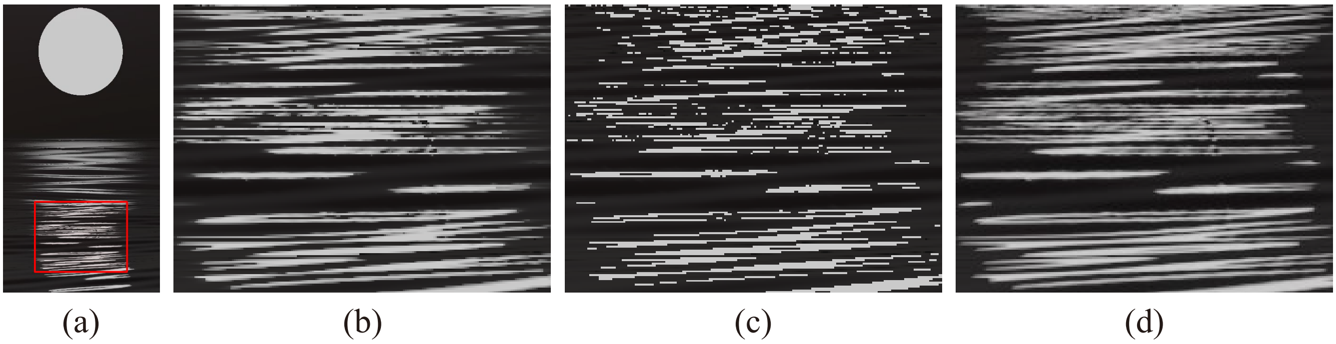

図 2 (a) The result of our method, (b) the closeup views of our method, (c) without anti-aliasing image, (d) reference image. The environment map "Milkyway" by Blochi ,via sIBL Archive (www.hdrlabs.com/sibl/archive.html)

York, NY, USA, ACM, pp. 289–296 (online), DOI: 10.1145/166117.166154 (1993).

[9] Norton, A., Rockwood, A. P. and Skolmoski, P. T.: Clamping: A Method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space, *Proceedings of ACM SIGGRAPH 1982*, pp. 1–8 (1982).

[10] Grant, C. W.: Integrated Analytic Spatial and Temporal Anti-aliasing for Polyhedra in 4-space, *SIGGRAPH Comput. Graph.*, Vol. 19, No. 3, pp. 79–84 (online), DOI: 10.1145/325165.325184 (1985).

[11] Zwicker, M., Pfister, H., van Baar, J. and Gross, M.: EWA Volume Splatting, *Proceedings of the Conference on Visualization '01*, pp. 29–36 (2001).

[12] Olano, M. and Baker, D.: LEAN Mapping, *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 181–188 (2010).