Computational Visual Media
https://doi.org/10.1007/s41095-021-0204-1 Vol. 7, No. 2, June 2021, 201–215
Research Article

Temporal and spatial anti-aliasing for rendering reflections on water waves

Namo Podee[1] (✉), **Nelson Max**[2], **Kei Iwasaki**[3], **and Yoshinori Dobashi**[1]

© The Author(s) 2021.

Abstract The reflection of a bright light source on a dynamic surface such as water with waves can be difficult to render well in real time due to reflection aliasing and flickering. In this paper, we propose a solution to this problem by approximating the reflection direction distribution for the water surface as an elliptical Gaussian distribution. Then we analytically integrate the reflection contribution throughout the rendering interval time. Our method can render in real time an animation of the time integrated reflection of a spherical light source on highly dynamic waves with reduced aliasing and flickering.

Keywords real-time rendering; anti-aliasing; reflection; water surface; water waves; elliptical Gaussian

1 Introduction

Ocean simulation and rendering have been studied in the computer graphics field for 40 years. That research can simulate an ocean under various weather conditions [1], and interacting with solid objects in three dimensions in real time [2]. However many problems in real-time ocean rendering remain. One of the problems is aliasing and flickering in rendering reflections on the water surface.

The ocean surface is highly dynamic. It moves rapidly and thus its shading changes rapidly as well. Usually, this does not pose any problems if the shading is smooth. However, for a surface that has a strong highlight or bright reflection moving rapidly, it causes inaccurate and unnatural flickering. In traditional rendering algorithms, each frame is rendered independently at discrete time steps, resulting in serious temporal aliasing artifacts. See Fig. 1 for an illustration of the problem. This paper focuses on this problem, assuming the light source is a sphere such as the full moon or the Sun.

Removing aliasing in real time is an active research area and many methods have been proposed [3–7]. 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 a water surface with waves.

In this paper, we present a method that can remove the spatial and temporal aliasing simultaneously, assuming that the water wave is represented as a sum of sine waves and the light source is a sphere.

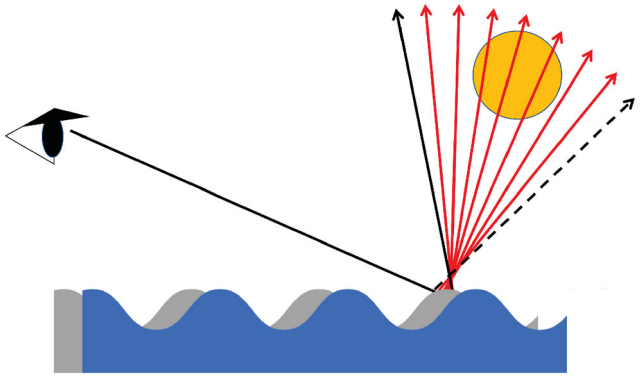


Fig. 1 The problem in rendering a reflection on a dynamic surface. Typically a single reflection ray (solid black) is generated for each frame, but it often misses the light source, causing temporal aliasing. We address this problem by also using the ray (dashed black) from the previous frame. The light source is sampled by using multiple rays (red) interpolating these two rays to reduce temporal aliasing.

1 Hokkaido University, Sapporo, 060-0814, Japan. E-mail: N. Podee, namo.podee@gmail.com (✉); Y. Dobashi, doba@ime.ist.hokudai.ac.jp.
2 University of California, Davis, 95616, USA. E-mail: max@cs.ucdavis.edu.
3 Wakayama University, Wakayama, 640-8441, Japan. E-mail: iwasaki@wakayama-u.ac.jp.
Manuscript received: 2020-11-02; accepted: 2021-01-07

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.

Our main contributions are:

- a method that analytically integrates a light reflection on a summed sine wave surface over a time period;
- a method that approximates a reflection direction distribution and then analytically integrates a light reflection on a summed sine wave surface over the reflection distribution and a time period;
- an improvement in accuracy for the Feline method of Ref. [8].

2 Related work

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

2.1 Reflection rendering

One of the most popular methods for rendering reflections in real time is image-based rendering. A method proposed in Ref. [9] renders realistic reflection images at an interactive speed by precomputing radiance maps that store the convolution 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 run time, McAuley et al. [10] introduced a split–sum approach that approximately computes the product of the environment map and the BRDF efficiently. More recent work achieves real-time shading for a polygonal light source [11] or a spherical light source [12]. However, none of these methods pay attention to temporal aliasing.

2.2 Screen space anti-aliasing

Fast approximate anti-aliasing (FXAA) [3] is one of the most popular screen space anti-aliasing methods. It simply detects jagged edges in an image and smooths them. It is fast but it blurs some details in the final image. Screen space temporal anti-aliasing has also been studied for a long time. One of the first methods was by Korein and Badler [4]. It determines the area of pixels that a moving object covers and then performs filtering on that area. This method works for a simple moving object. A method by Shinya [13] uses a velocity field to track surfaces moving on a screen. Then the method reconstructs an anti-aliased result by blending pixels with the same surface position in multiple frames. This method can be called a temporal reprojection method. It has a wide variety of applications [14, 15], but suffers from three main problems. Firstly, it cannot deal with any surface that is hidden in previous frames. Secondly, it has parameters that needed to be adjusted carefully. Thirdly, it cannot track the movement of the reflection direction, which is our main objective.

2.3 Analytic anti-aliasing

Our method belongs to this group. Such methods analyze the cause of the aliasing and find a mathematical solution for each target aliasing situation. A clamping method [5] removes aliasing on a textured surface by using the Nyquist theorem [16]; we use it in our method. The method proposed in Ref. [17] analyzes the movement of a polygon in image space and generates a space–time representation of the object for spatial and temporal anti-aliasing. EWA volume splatting [18] is an anti-aliasing method for volume rendering. It uses an elliptical Gaussian function as a reconstruction kernel for the volume to avoid aliasing. We are inspired by this work and borrow the idea of its Gaussian kernel's affine mapping. Our method also builds on LEAN mapping [19] that approximates the distribution of surface normal vectors with a Gaussian function to eliminate aliasing that comes from using a bump map. LEADR mapping [6] is another method that extends LEAN mapping to cover physically based rendering. Becker and Max [20] combined three rendering algorithms, based on BRDF, bump mapping, and displacement mapping, and transition between them in an aliasing-optimal way. Bruneton et al. [7] also used these rendering algorithms as three levels of detail to eliminate aliasing on a dynamic water surface in real time. Wu et al. [21] accurately rendered anti-aliased materials across multiple levels of detail by generating an SVBRDF from a displacement map and a BRDF. The method considers occlusion, shadow-masking, and inter-reflection effects of the displacement map as well. However, these methods do not take into account temporal aliasing, which we address in this

paper. Feline mapping [8] also helps us improve the efficiency and quality of our work. It is a texture filtering method that approximates an anisotropic filter as the sum of multiple isotropic filters.

Less related to our work is Monte Carlo distributed ray tracing [22], which combines spatial and temporal aliasing by tracing many rays from random points in the finite pixel area, and at random time in the frame interval. It produces noisy images unless very many rays are used, but modern noise removal algorithms can decrease this noise fairly effectively.

3 Proposed method

3.1 Overview

Our goal is to compute the contribution of a spherical light source over the period of time between frames. As mentioned earlier, 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 spatial frequency bands of low- and high-frequency, respectively temporal-aliasing waves and spatio-temporal-aliasing waves. The high-spatial-frequency component causes spatial aliasing due to under-sampling and may also cause temporal aliasing due to its movement. We give two methods, temporal-aliasing wave rendering and spatio-temporal-aliasing wave rendering, for these two frequency bands, respectively, and make a smooth transition between them. The fundamental idea of both methods is the same for each sine wave, but in the second method, the spherical light source is blurred to account for the effects of the BRDF caused by the aliasing waves. We also using a height field of the wave as a displacement map of our water surface geometry. Figure 2 outlines our method.

3.2 Spatial-aliasing detection

We use the clamping anti-aliasing method [5] to decompose the water wave into the spatio-temporal-aliasing and temporal-aliasing waves. According to the Nyquist theorem from sampling theory, to avoid aliasing, the spatial sampling frequency must be higher than twice the highest frequency of the water wave. In our case, this means that the pixel resolution should be twice as fine as the wavelength of the water wave projected onto the screen. Thus, when a wave is projected onto the screen, the wave must cover at least

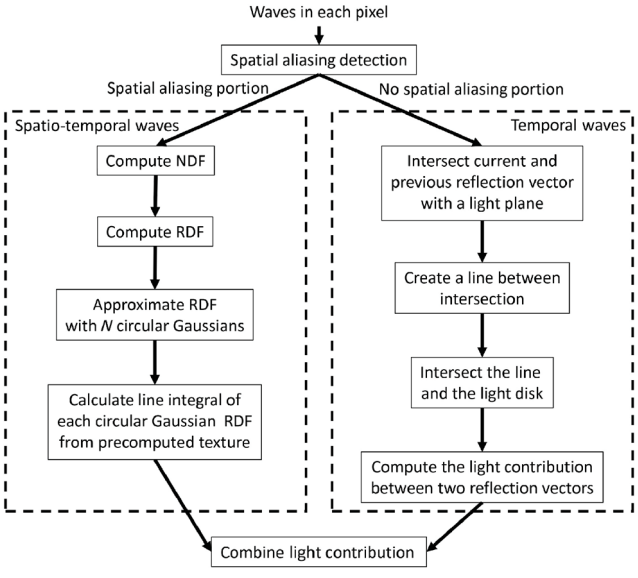


Fig. 2 Outline of our methodology.

two pixels within one wavelength. This maximum frequency of the projected wave is called the Nyquist limit. If a wave does not meet this criterion, then it will cause aliasing. The maximum frequency, or the minimum wavelength, happens when the wave is aligned horizontally or vertically with our screen space as can be seen in Fig. 3. Thus, our Nyquist limit is 1/2, which is the spatial frequency of a wave with a wavelength equal to the size of two pixels.

For each pixel, we calculate the projected wavelength of each sine wave on the screen. Each sine wave is then classified per pixel as either temporal-aliasing (called simply "non-aliasing" below) or spatio-temporal-aliasing (called "aliasing" below) according to the Nyquist frequency, which detects spatial aliasing. Pixels that only contain non-aliasing waves and pixels that contain aliasing waves will be processed by two different rendering methods, which produce slightly different results.

A sudden transition between the two types of waves

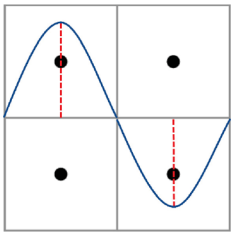


Fig. 3 A projected wave aligned to a 2×2 screen. Any wave in any projected direction with higher spatial frequency or lower wavelength than this wave will cause aliasing.

will create arcs of sudden appearance change on the boundary between pixels with and without aliasing waves. To prevent this, we use soft classification to smoothly blend the results from both methods. Section 3.7 explains the blending method in detail.

3.3 Temporal-aliasing wave rendering

For a pixel that only contains non-aliasing waves, we sample a single point on the water surface corresponding to the pixel center and compute the contribution of the reflected light over the time interval between the current frame and previous frame. We assume that the spherical light source is distant from the sample point so that it subtends the same solid angle from every sample point, and is represented by a disk facing the sample point. This disk is the projection of the light sphere onto a light plane perpendicular to the line from the sample point to the center of the light sphere. Two reflection vectors are computed by using the normal vectors at the previous and the current frames. We then calculate the intersection points between the light disk and a plane formed by the two reflection vectors, as shown in Fig. 4. The light disk contribution between these two reflection vectors can be calculated as follows. First, we calculate two intersection points p_c and p_p between the two reflection vectors and the plane. Next, we calculate the overlapping line segment between $p_c p_p$ and the light disk. Then the arc subtended by $p_c p_p$ (the whole arc) and that subtended by the overlapping line segment (the light arc) is calculated. The light source's fractional contribution between the successive frames is obtained as the ratio of the angle of the light arc to that of the whole arc, as shown in Fig. 4.

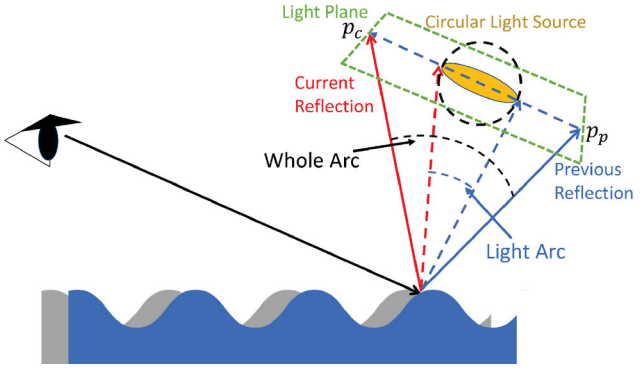


Fig. 4 Our method uses simple geometric analysis to find the contribution of light between the current reflection and the previous reflection.

3.4 Spatio-temporal-aliasing wave rendering

For a pixel that contains any aliasing waves, a single sample point per pixel is insufficient. We have to compute the average intensity of the reflected light over the pixel area taking into account the distribution of surface normals. A straightforward solution is to generate multiple rays for each pixel as in Ref. [22], which significantly increases the computation time. Instead, we borrow the idea of the LEAN mapping technique [19] for efficient computation, which approximates the distribution by a Gaussian distribution. However, this approximation removes details from the surface and blurs the reflection result, which makes temporal reflection changes less noticeable. Thus, our method assumes that aliasing waves are unchanging in time and only non-aliasing waves contribute temporal changes to the pixel's reflection result. For a pixel that contains both types of waves, we need to address spatial aliasing from aliasing waves and temporal aliasing from non-aliasing waves. To eliminate spatial aliasing, our method approximates an aliasing wave as a distribution function of its surface normal. This distribution itself does not change over time, but it is translated by the normal of the non-aliased waves, according to our assumptions. Our method then removes temporal aliasing by computing the time-integrated reflection of the distribution and the light source. Figure 5 shows the outline of our spatio-temporal-aliasing wave rendering method.

We first calculate the normal distribution function (NDF) of the aliasing waves. The NDF is defined as a distribution of normal vector on a horizontal x–z plane, assuming the y axis points in the upward

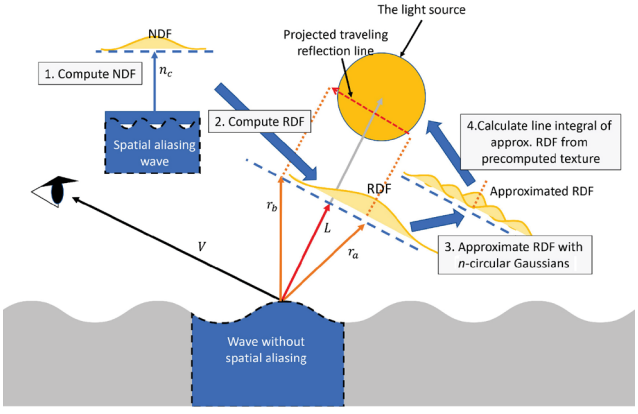


Fig. 5 Outline of our spatio-temporal wave rendering method.

direction (see Fig. 6). We call this the NDF plane. The NDF for the sum of the sine waves is obtained by accumulating the NDF for each sine wave. So, let us consider the NDF for the following single sine wave:

$$y(x,z,t) = A\sin(ax+bz+ct) \tag{1}$$

where A is the amplitude and t is the time. The wave vector (a,b) determines the wavelength and direction of the wave, and c together with the wave vector determines the velocity. We approximate the NDF by a two-dimensional elliptical Gaussian function of a point $\boldsymbol{n}$ on the NDF plane:

$$\mathcal{G}_{\Sigma,\boldsymbol{n}_c}(\boldsymbol{n}) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}}\mathrm{e}^{-\frac{1}{2}(\boldsymbol{n}-\boldsymbol{n}_c)^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{n}-\boldsymbol{n}_c)} \tag{2}$$

where $\boldsymbol{n}_c$ is the point corresponding to the normal vector without aliasing waves (i.e., the low frequency components) and is the center of the Gaussian on the NDF plane. Σ is the covariance matrix of the elliptical Gaussian and can be expressed differently by our two approximation methods, the uniform method and the per-pixel method.

Our uniform method assumes that each aliasing wave has its whole wave cycle inside each pixel, so its Σ can be expressed as

$$\Sigma = \lim_{T\to\infty}\frac{1}{T}\begin{bmatrix}\int_0^T\left(\frac{\partial y}{\partial x}\right)^2\mathrm{d}t & \int_0^T\frac{\partial y}{\partial x}\frac{\partial y}{\partial z}\mathrm{d}t\\ \int_0^T\frac{\partial y}{\partial x}\frac{\partial y}{\partial z}\mathrm{d}t & \int_0^T\left(\frac{\partial y}{\partial z}\right)^2\mathrm{d}t\end{bmatrix}$$
$$= \frac{1}{2}\begin{bmatrix}A^2a^2 & A^2ab\\ A^2ab & A^2b^2\end{bmatrix} = \begin{bmatrix}\sigma_x^2 & \sigma_x\sigma_z\\ \sigma_x\sigma_z & \sigma_z^2\end{bmatrix} \tag{3}$$

where $\sigma_x = Aa/\sqrt{2}$ and $\sigma_z = Ab/\sqrt{2}$, and the temporal average over t is the same as the spatial average over x and z because of the form of Eq. (1).

Our per-pixel method finds the start and end of the sine wave within the pixel, where Σ can be defined more accurately at the cost of increased computation. The covariance matrix Σ can be instead expressed by

$$\Sigma = \begin{bmatrix}M.xx - B.x\times B.x & M.xz - B.x\times B.z\\ M.xz - B.x\times B.z & M.zz - B.z\times B.z\end{bmatrix} \tag{4}$$

$$B = (B.x, B.z) = \frac{1}{p_2-p_1}\left(\int_{p_1}^{p_2}\frac{\partial y}{\partial x}\mathrm{d}p, \int_{p_1}^{p_2}\frac{\partial y}{\partial z}\mathrm{d}p\right) \tag{5}$$

$$M = (M.xx, M.xz, M.zz) = \frac{1}{p_2-p_1}$$
$$\cdot\left(\int_{p_1}^{p_2}\left(\frac{\partial y}{\partial x}\right)^2\mathrm{d}p, \int_{p_1}^{p_2}\frac{\partial y}{\partial x}\frac{\partial y}{\partial z}\mathrm{d}p, \int_{p_1}^{p_2}\left(\frac{\partial y}{\partial z}\right)^2\mathrm{d}p\right) \tag{6}$$

$$p(x,z) = ax + bz \tag{7}$$

where p_1 and p_2 are the start and end of the wave within the pixel respectively. However, p_1 and p_2 of each wave within the pixel vary with position. To simplify the problem, we only consider the position at the pixel center. Equations (5) and (6) can be solved analytically, but we omit the detail here to save space. Appendix A gives the solution to the equations and explains how we find p_1 and p_2 in more detail. The NDF for the sum of the waves is approximated as the sum of the individual wave NDFs.

The NDF computed above is then transformed into reflection space to obtain the reflection distribution function (RDF), which represents the distribution of the reflection directions. The RDF is defined on a light plane perpendicular to the unit direction vector to the center of the light source and is located at the endpoint of this unit vector, as shown in Fig. 6. We explain the derivation of our transformation function below (see also Fig. 7).

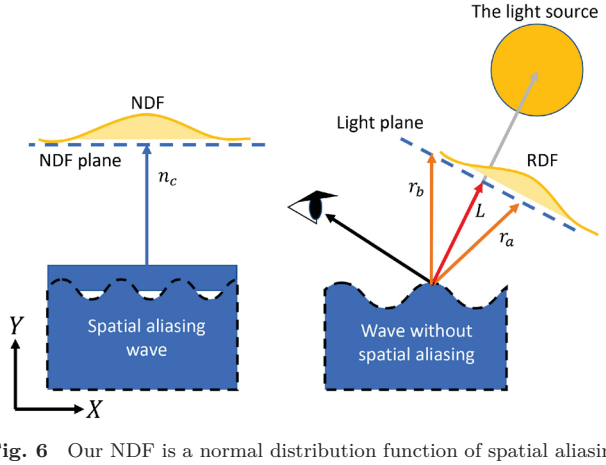


Fig. 6 Our NDF is a normal distribution function of spatial aliasing waves. Our RDF is a reflection distribution function of the NDF. It is defined on a light plane perpendicular to the direction to the center of the light source.

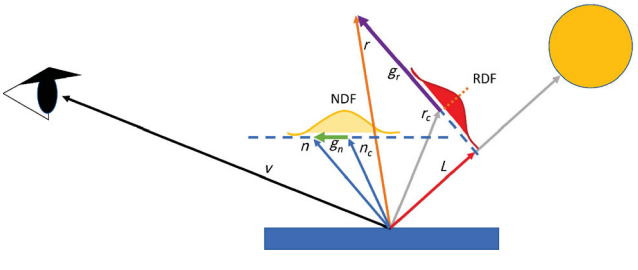


Fig. 7 Transformation from NDF to RDF.

Let us consider a function Φ that maps point $\boldsymbol{n}$ on the NDF plane to point $\boldsymbol{r}$ on the light plane, i.e., $\boldsymbol{r} = \Phi(\boldsymbol{n})$. $\Phi(\boldsymbol{n})$ can be derived by geometrical analysis of Fig. 7, as follows:

$$\Phi(\boldsymbol{n}) = \frac{r(\boldsymbol{n})}{r(\boldsymbol{n}) \cdot \boldsymbol{L}} \tag{8}$$

$$r(\boldsymbol{n}) = 2(\hat{\boldsymbol{n}} \cdot \boldsymbol{v})\hat{\boldsymbol{n}} - \boldsymbol{v} \tag{9}$$

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{n}}{|\boldsymbol{n}|} \tag{10}$$

where $\boldsymbol{v}$ is the eye direction and $\boldsymbol{L}$ is the light direction. As Eqs. (8)–(10) show, Φ is a complicated nonlinear function, so we employ a first order approximation. Let $\boldsymbol{n}_c$ be the center of the NDF, $\boldsymbol{r}_c = r(\boldsymbol{n}_c)$ be the center of the RDF, and $\boldsymbol{g}_n = \boldsymbol{n} - \boldsymbol{n}_c$. Then the approximation is

$$\Phi(\boldsymbol{n}) \approx \Phi(\boldsymbol{n}_c) + J_c \boldsymbol{g}_n = \boldsymbol{r}_c + J_c \boldsymbol{g}_n \tag{11}$$

$$J_c = \frac{\partial \varphi}{\partial \boldsymbol{g}_n}(0) \tag{12}$$

$$\varphi(\boldsymbol{g}_n) = \boldsymbol{g}_r = \boldsymbol{r} - \boldsymbol{r}_c = r(\boldsymbol{g}_n + \boldsymbol{n}_c) - [2(\boldsymbol{n}_c \cdot \boldsymbol{v})\boldsymbol{n}_c - \boldsymbol{v}] \tag{13}$$

where J_c is the Jacobian matrix of $\varphi(\boldsymbol{g}_n)$ computed at $\boldsymbol{g}_n = 0$. $\varphi(\boldsymbol{g}_n)$ is a function that maps point $\boldsymbol{g}_n$ on the NDF plane to point $\boldsymbol{g}_r = \boldsymbol{r} - \boldsymbol{r}_c$ on the light plane. Then, Φ^{-1} may be approximated by

$$\Phi^{-1}(\boldsymbol{r}) \approx \boldsymbol{n}_c + J_c^{-1} \boldsymbol{g}_r \tag{14}$$

With this linear approximation, the RDF can also be expressed by another elliptical Gaussian function with covariance matrix Σ_r. That is, the RDF is expressed as

$$\begin{aligned} \mathrm{RDF}(\boldsymbol{r}) &= \mathrm{NDF}(\Phi^{-1}(\boldsymbol{r})) \\ &\approx \frac{1}{|J_c^{-1}|} \mathcal{G}_{\Sigma_r, \boldsymbol{r}_c}(\boldsymbol{r}) \end{aligned} \tag{15}$$

$$\Sigma_r = J_c \Sigma J_c^{\mathrm{T}} \tag{16}$$

Note that $\boldsymbol{r}_c$ is the center of the above Gaussian function on the light plane. The accuracy of our RDF approximation is considered in Appendix B.

We can now compute the intensity of the reflected light using the convolution of the RDF and the light disk which is the projection of the spherical light source onto the light plane. However, to remove temporal aliasing, we need to compute the average intensity over the time interval between the frames. Thus, the intensity I is given by

$$I = \frac{L_l}{T} \int_{t-T}^{t} \int_{\boldsymbol{r} \in D} \mathrm{RDF}(\boldsymbol{r}, t^*)\, \mathrm{d}\boldsymbol{r}\, \mathrm{d}t^* \tag{17}$$

where L_l is the radiance of the light source and T is the time interval between the frames. D represents the circular region within the light disk. Note that we include the time parameter t^* for the RDF to explicitly indicate its time-variance.

The above equation is too expensive to compute in real time. We therefore precompute the integral under some simplifying assumptions. We assume that the path of the reflection direction on the light plane during the frame is a straight line segment. The accuracy of this approximation can be improved by dividing the frame time into subintervals, each with its own straight segment. We also approximate the elliptical Gaussian RDF by a circular Gaussian, with variance $\sigma_r^2 = \sigma_s \sigma_t$, where σ_s and σ_t are the eigenvalues of the covariance matrix Σ_r in Eq. (15). Later we approximate the elliptical Gaussian more accurately as a weighted sum of circular Gaussians. Finally, we assume that temporal changes in the Jacobian J_c and covariance matrix Σ_r are small. Thus, J_c and Σ_r are also assumed to be constant between neighboring frames. We only consider the change in the primary normal direction $\boldsymbol{n}_c$ (and therefore the primary reflection direction $\boldsymbol{r}_c$, i.e., the center of the Gaussian RDF).

3.5 Circular Gaussian approximation

For the circular Gaussian, we need to precompute Eq. (17), which is an integral, over the reflection vector segment for the frame time, of the RDF convolved with the light disk. Because this convolution is circularly symmetric, the integral can be precomputed using three parameters: the closest distance of the line containing the segment to the center of the convolution, the start position of the segment on that line, and the end position. See Fig. 8. Considering also the variance σ_r^2 for the RDF, this gives four parameters.

Fortunately, we can reduce these to three parameters. The straight segment integration can be separated into four cases: a segment that starts and ends at the outside but cuts through the convolution, a segment that starts at the outside but ends inside the convolution, which, by reflection symmetry across the V axis in Fig. 8, also includes the segment which starts inside and ends outside, a segment that starts and ends inside the convolution, and a segment that starts and ends outside the convolution without cutting through it, which has integral zero. The first case has one parameter, which is the closest distance of the segment to the center of the convolution.

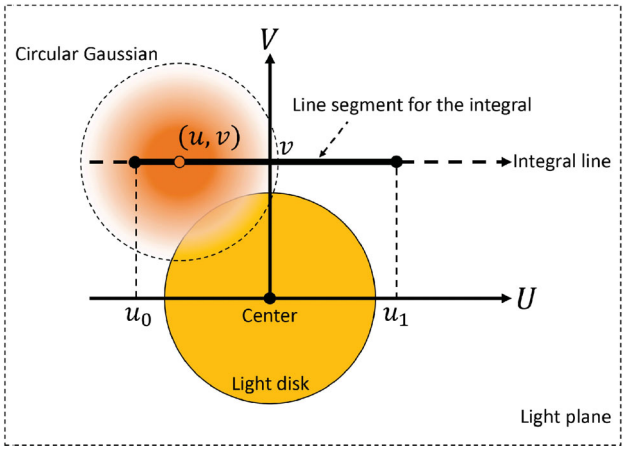


Fig. 8 Precomputation of the line integral, which is a convolution of the light disk and the circular Gaussian that is moving on the line segment.

The second case has two parameters, which are the closest distance of the segment to the center of the convolution and the end position of the segment. The third case has the three parameters noted before. However, we can calculate the third case integral by first finding the integral over a segment that starts from an infinite distance and goes to the end position while passing through the start position of the integral segment. Then, we subtract another integral on the same line that starts from an infinite distance and goes to the start position of the segment. In other words, an integral of the third case can be calculated as the difference of two second case integrals. Thus, we only need to precompute the first and second cases, and the first is a special case of the second. The second case needs two parameters, which are the closest distance and the end position of the segment. Thus, when including the variance too, we can store the line integral in a 3D texture, as described below.

Let us consider local coordinates UV on the light plane with origin at the center of the light disk (see Fig. 8). Because of circular symmetry, we can assume that the line segment for the line integral is parallel to the U axis and intersects the V axis at $V = v$, shown as a thick black line segment in the figure. The center of the circular Gaussian RDF moves along the integral line. Our method precomputes the following table for various variances.

$$S(u, v, \sigma_r) = \int_{-R}^{u} F(u', v, \sigma_r) \mathrm{d}u', \quad -R < u, v < R \tag{18}$$

where $F(u, v, \sigma_r)$ corresponds to the inner integral of Eq. (17), i.e., the convolution of the light disk with the circular Gaussian whose center is at (u, v). R is a large value chosen such that $F(R, 0, \sigma_r)$ is sufficiently small. With this table, the intensity expressed by Eq. (17) is obtained by

$$I = S(u_1, v, \sigma_r) - S(u_0, v, \sigma_r) \tag{19}$$

where (u_0, v) and (u_1, v) correspond to the two primary reflection directions for the previous and current frames, exchanged if necessary to make $u_1 \geqslant u_0$. Our method stores S in a 3D texture on the GPU. The precomputation only considers a constant light disk radius, but we can linearly transform the light disk radius and the RDF variance together. Thus, we can transform various input parameters to fit our precomputed data before using it. Our method supports changing both light and wave parameters in real time.

3.6 Elliptical Gaussian approximation

We needed to approximate an elliptical Gaussian RDF as a circular Gaussian to keep the precomputed table to 3 dimensions, to be able to use hardware support on a GPU. However, this approximation loses accuracy when an RDF has a very elongated distribution along one of its axes, which happens when the viewing direction is close to the horizon. So we approximate an elliptical Gaussian RDF more closely with a method similar to Feline mapping [8], which uses multiple circular Gaussian kernels to approximate a single elliptical Gaussian kernel. However, we improve the method in Ref. [8] by non-linearly optimizing the circular Gaussian kernel weights, center positions, and variances so that their combination matches the elliptical Gaussian RDF kernel as closely as possible. Figure 9 illustrates the idea. Details of this fit are given in Appendix C, and produce curve fits for the circular Gaussian parameters.

When we are rendering the scene, after we obtain the RDF, we scale the whole RDF so that its minor axis variance is equal to 1. Then we find the optimal circular Gaussian kernel parameters by inserting the scaled major axis variance of the RDF into each curve-fitted equation of each parameter as a function of the major axis variance. Next, we scale the optimal parameters back to the original scale. Then we find the line segment of each circular Gaussian kernel by offsetting the segment from the original segment by a vector from the center of the RDF to the center of each circular Gaussian kernel. Figure 10 shows an

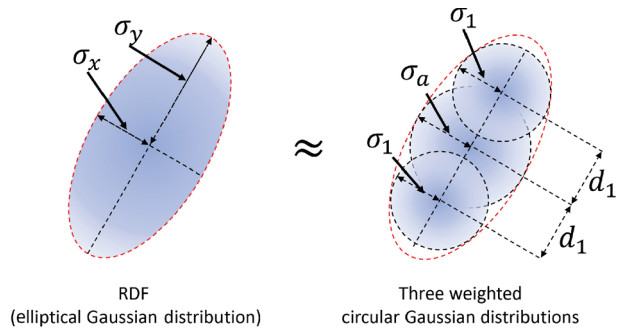


Fig. 9 We further approximate our RDF, which is defined as an elliptical Gaussian distribution, as a combination of multiple weighted circular Gaussian distributions. In this example, we approximate the RDF with 3-weighted circular Gaussian distributions.

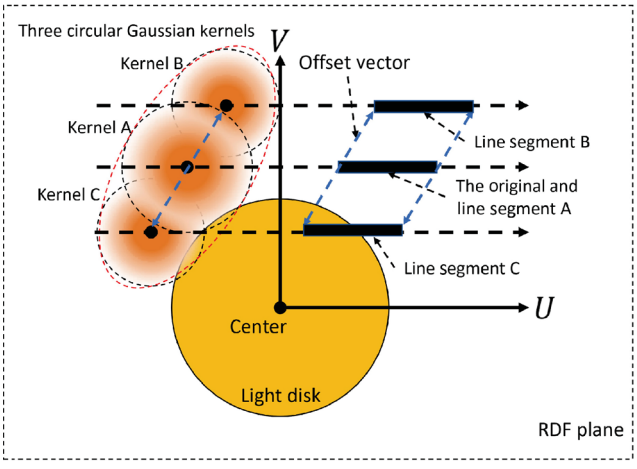


Fig. 10 Three circular Gaussian kernels are used to approximate the RDF. Each circular Gaussian kernel has its own integral segment, which can be calculated by offsetting the original integral segment by the vector from the center of the RDF to the circular Gaussian kernel.

example of line segment offsetting for a case with three circular Gaussian kernels. Finally, we use our circular approximation method to find the result with the adjusted segment for each circular Gaussian kernel, and combine them according to the unscaled optimal weight parameters.

However, there is a limitation to this method. An elliptical Gaussian kernel with a more elongated shape will require more circular Gaussian kernels, which increases computational time. To have stable performance, we use a fixed number of circular Gaussian kernels (nine in our implementation.) Then if the elongation (ratio of major to minor axis) is more than a threshold, we use our fallback method instead. The threshold value we use is the standard deviation of the minor axis of the RDF multiplied by the number of circular Gaussian kernels.

Our fallback method is computing a convolution of the RDF and the light source by the Riemann sum method. We ignore the temporal integration in our fallback method because the reflection from a high variance RDF is blurry, so is less likely to be improved by temporal integration.

3.7 Aliasing wave transition

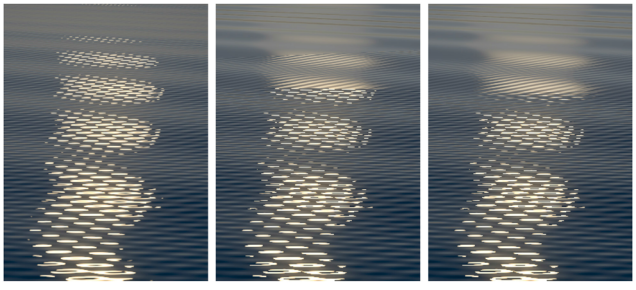
Our method treats non-aliasing waves and aliasing waves differently. The non-aliasing waves are used to determine temporal changes of the surface, while aliasing waves are approximated by a Gaussian RDF. However, this approximation, which is in our spatio-temporal-aliasing wave rendering method, causes visual artifacts in our result. When we use the Nyquist limit to classify the type of each wave within each pixel, it causes a differently rendered result for the two types of waves. This switch is too sudden. Any wave with a projected frequency lower than the Nyquist limit is considered a non-aliasing wave, while the same wave in a neighbor pixel might have projected frequency equal or slightly above the Nyquist limit and be suddenly considered an aliasing wave. The difference between the two pixels will be apparent as seen in Fig. 11(b).

To prevent this, instead of using the Nyquist limit, we use soft classification to smoothly go from non-aliasing to aliasing waves. We create a transition region, with a fractional aliasing value α between 0 and 1. The transition region is in frequency space and stops before reaching the Nyquist limit. Any wave within the region is considered to contain both aliased and non-aliased waves with weights of α and $1-\alpha$, respectively. If the transition region has a width of δ in frequency space and the Nyquist limit is f_n, the transition region starts at $f_n-\delta$ and ends at f_n. A wave with a frequency of f within the region has a transition value α given by

$$\alpha = \begin{cases} 0, & f < f_n - \delta \\ [f-(f_n-\delta)]/\delta, & f_n - \delta \leqslant f \leqslant f_n \\ 1, & f_n < f \end{cases} \tag{20}$$

For α between 0 and 1, we transform the sine wave into two waves, an aliased wave with amplitude weighted by α, and a non-aliased wave weighted by $1-\alpha$. We do this for every sine wave, and combine waves of the same type. If there are no aliased waves, the combined non-aliased waves will be rendered with our temporal-aliasing wave rendering method. However, if there are aliased waves, the combined

aliased waves will be represented as an NDF and the combined non-aliased waves will determine a reflection path in our spatio-temporal-aliasing wave rendering method. Figure 11 shows an example transition.



(a) Reference (b) No transition (c) with transition

Fig. 11 With and without transition regions.

4 Implementation

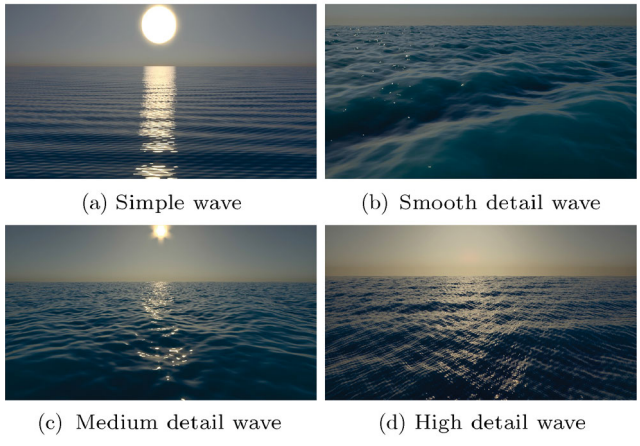
We implemented our method with OpenGL. The size of the transition region between aliasing and non-aliasing waves can be adjusted to achieve a smooth transition between them. Our temporal-aliasing rendering method is faster than our spatio-temporal-aliasing wave rendering method. Thus, we need to adjust the transition region to use the temporal-aliasing wave rendering method as much as possible. We experimentally determined the transition region to start at 0.2 cycles per pixel and finish at 0.5 cycles per pixel.

In our implementation of spatio-temporal-antialiasing, the size of the precomputed 3D texture for the circular Gaussian approximation is $512 \times 512 \times 64$; the dimensions correspond to the closest distance, line segment endpoint, and variance of the circular Gaussian kernel, respectively. For the elliptical Gaussian approximation, we use 1, 3, 5, or 9 circular Gaussian kernels. We provide curve fitting results for each of these in Tables S1–S3 in the Electronic Supplementary Material (ESM). However, we cannot use this 3D texture when there are no non-aliased waves that vary in time because we cannot find the point integral of a still reflection vector with our method (there is no reflection path). Thus, we need to precompute another 2D texture that just stores a convolution result of a circular Gaussian kernel with a light disk with a resolution of 512×64; each dimension corresponds to the distance between the center of the light disk and the kernel, and variance of the circular Gaussian kernel, respectively.

5 Results

We investigated the abilities of our method by using four scenes with a simple wave, a smooth detail wave, a medium detail wave, and a high detail wave (see Fig. 12). For each scene, we created multiple images using six different rendering methods: without anti-aliasing, spatial-only 32 multisample anti-aliasing (32x MSAA), LEAN mapping, our method with uniform NDF, our method with per-pixel NDF, and a reference image. We generated a 4K normal map from sine wave data for LEAN mapping for each frame. Our method uses nine circular Gaussian kernels for the temporal-aliasing-wave rendering method. The reference image was created offline by simply generating a supersampled image with 2×2, 4×4, or 8×8 higher resolution and then downsampling it. We also included temporal integration in the offline rendering by generating 4 or 8 supersampled images between neighboring frames. Glare and tone mapping post-processing were applied to all images, to show the Sun's high brightness reflection. The rendering time is shown in the captions of the figures; our precomputation time is 4.24 ms. The rendering time is measured on a PC with Intel Core i9-9900KF 3.6 GHz CPU, 16 GB RAM, and NVIDIA TITAN RTX GPU with 24 GB RAM (also see the accompanying video animations in the ESM).

Figure 13 shows a comparison using a simple wave consisting of five sine waves with similar directions. This scene shows a simple and predictable reflection movement. Our method produces almost the same



(a) Simple wave (b) Smooth detail wave

(c) Medium detail wave (d) High detail wave

Fig. 12 Our four experimental scenes.

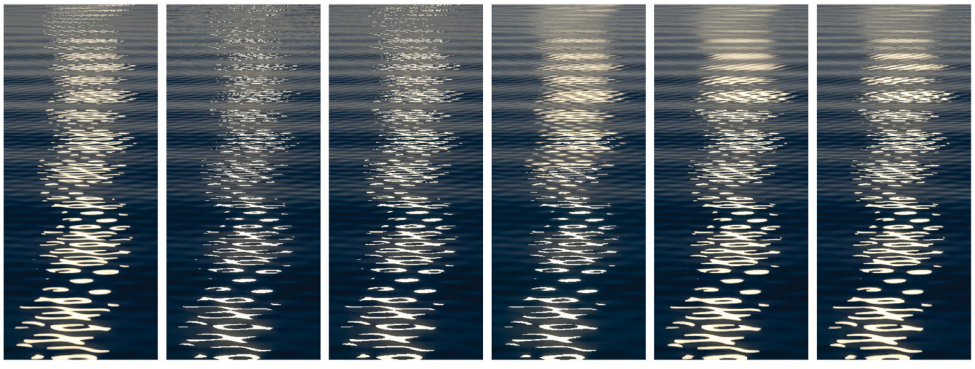


Fig. 13 Reflection result for the simple wave scene. The reference uses 8 spatial samples and 4 temporal samples.

result as the reference image for regions near the camera. In the far region, our elliptical Gaussian approximation removes the spatial aliasing. However, the other two methods produce severe spatial aliasing in both near and far regions.

Figures 14 and 15 use smooth and medium detail waves, consisting of 64 and 48 sine waves, respectively. These waves exhibit sparse reflections that are difficult for previous methods to capture. Our method successfully renders an accurate image.

Figure 16 shows images rendered with a high detail wave consisting of 32 sine waves of shorter wavelength. Again, our method can render an accurate reflection image near the camera while spatial aliasing is removed in the far region. The other two methods suffer from severe aliasing.

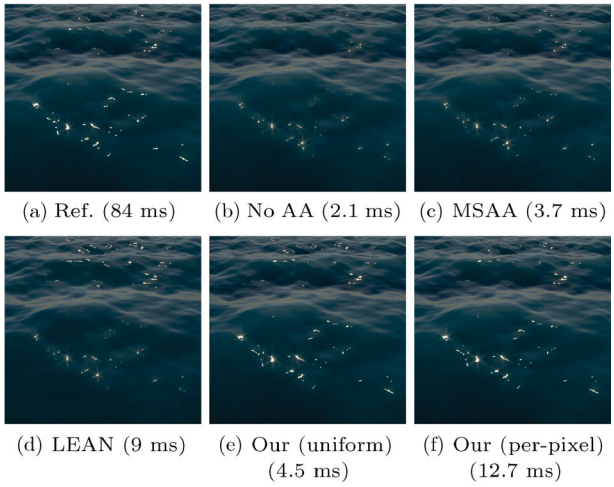


Fig. 14 Reflection result for the smooth detail wave scene. The reference uses 4 spatial samples and 8 temporal samples.

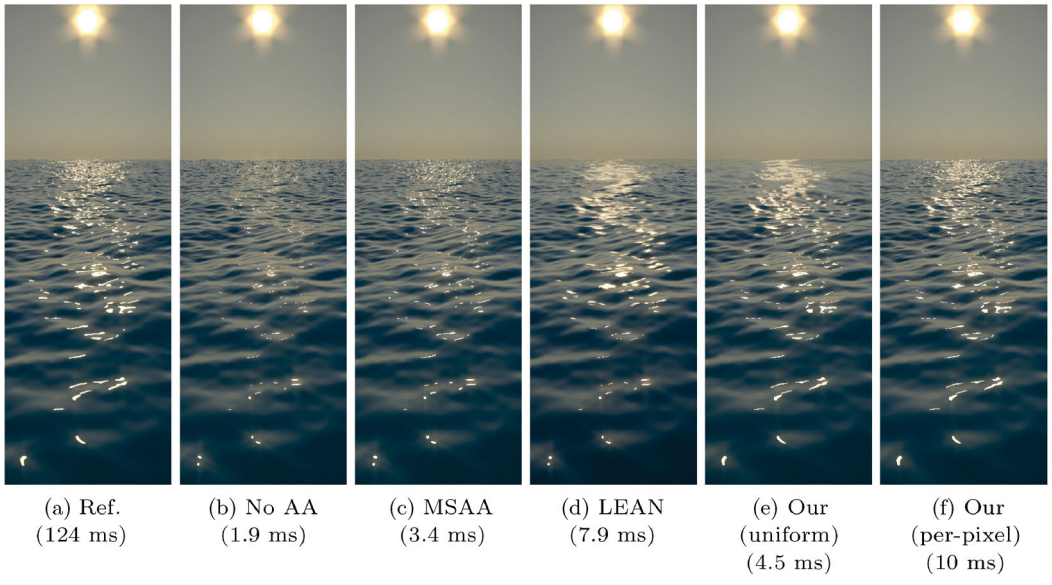
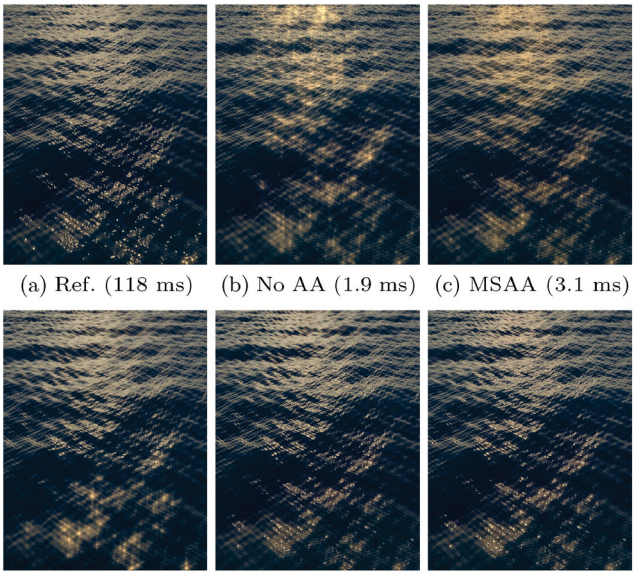


Fig. 15 Reflection result for the medium detail wave scene. The reference uses 8 spatial samples and 8 temporal samples.



(a) Ref. (118 ms) (b) No AA (1.9 ms) (c) MSAA (3.1 ms)

(d) LEAN (6.7 ms) (e) Our (uniform) (5.3 ms) (f) Our (per-pixel) (12.1 ms)

Fig. 16 Reflection result for the high detail wave scene. The reference uses 8 spatial samples and 8 temporal samples.

Table 1 and Fig. 18 show that our method produces results closer to the references than other methods.

Table 1 RMSE of the result of each method compared to the reference

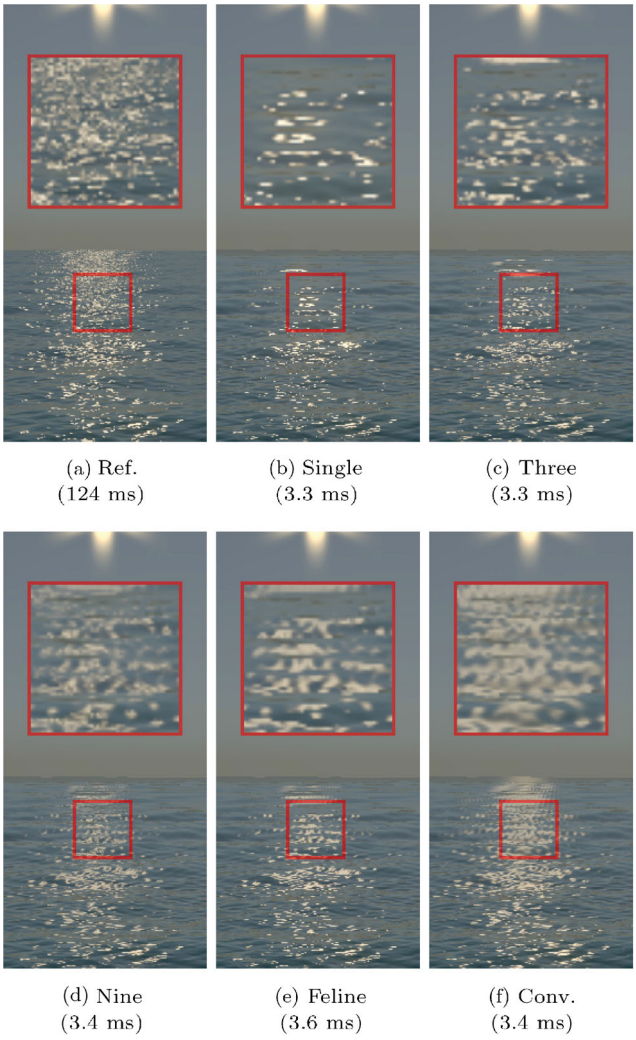
	Scene			
Method	Simple	Smooth	Medium	High
No AA	0.0797	0.0299	0.0315	0.0778
32x MSAA	0.0795	0.0287	0.0282	0.0676
LEAN	0.0675	0.0329	0.0298	0.0633
Ours (uniform)	**0.0384**	0.0200	0.0278	0.0611
Ours (per-pixel)	**0.0384**	**0.0192**	**0.0229**	**0.0610**

6 Discussion

Our method particularly improves rendering of reflections near the camera as shown in the examples in the previous section. However, images of the reflection in regions far from the camera are blurred due to our elliptical Gaussian approximation. This problem is reduced by using per-pixel NDF at the cost of more computational time. Per-pixel NDF does not reduce the error as much as we expected, but it produces a sharp reflection result, which is visually similar to the reference.

Our elliptical Gaussian approximation increases the accuracy of the elongated reflection shape as we increase the number of circular Gaussian kernel. According to our experiments, three circular Gaussian kernels are enough, as shown in Fig. 17 but increasing the number of kernels can improve the quality of reflection at the horizon with little increased computational effort. In Fig. 17, we also include an accurate convolution result of the Gaussian-approximated RDF with a light source, which is computed by using the Riemann sum method, for easier comparison.

This paper does not consider the effects of occlusion (from the viewpoint) or shadowing (from the light source) of one wave by another. Smith [23] developed equations for these, which were correctly normalized by Ross [24], under the assumption that the waves



(a) Ref. (124 ms) (b) Single (3.3 ms) (c) Three (3.3 ms)

(d) Nine (3.4 ms) (e) Feline (3.6 ms) (f) Conv. (3.4 ms)

Fig. 17 Comparison of our approximation using different numbers of circular Gaussian kernels (without a fallback method), the Feline method, and convolution. The reference uses 8 spatial samples. The RMSEs of (b)–(f) are 0.0209, 0.0195, 0.0181, 0.0191, and 0.0183 respectively. (d) has the best performance for capturing and ability to compute time integral but (f) is better at capturing the reflection near the horizon. Thus, our method uses both (d) and (f).

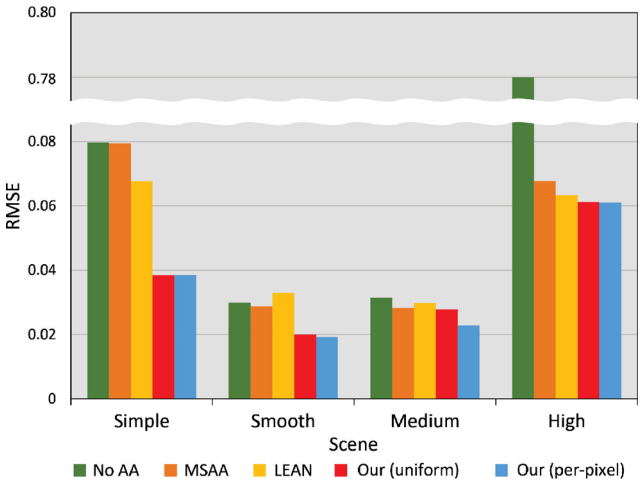


Fig. 18 RMSE comparison.

were random and that the normals were uncorrelated with the wave height. These ideas were applied to rendering sea waves by Bruneton and Neyret [7], and to general surfaces by Dupuy et al. [6], enhancing LEAN mapping to LEADR mapping. They could be combined with our temporal anti-aliasing. However, when summing only a few sine waves, the heights and normals are in fact correlated. Becker and Max [20] handled visibility at the mesoscale where normal perturbation replaces displacement mapping, by precomputing a redistributed normal map which accounted for occlusion, but not shadowing; this method can apply to any height field. However, it requires precomputation of visibility from any viewing angle, which can not currently be done in real time for a time-varying height field.

7 Conclusions and future work

Our method reduces aliasing and increases the fluidity of light disk reflection animation on a water surface by using spatio-temporal anti-aliasing. Our method analytically solves the aliasing problem and thus it does not have limitations like the temporal reprojection method [14, 15] that suffers from hidden-surface and reflection tracking problems.

Our main current limitation is that our method cannot replicate small details of distant reflections. This is because we approximate the NDF and the RDF by Gaussian distributions. The problem becomes more apparent when we use a uniform NDF, which calculates a single average NDF for the entire ocean wave. The problem is reduced when we use per-pixel NDF.

In future, we plan to improve our method to consider multiple frame temporal anti-aliasing, and other types of light sources and waves. Multiple frame temporal anti-aliasing will simulate the effect of longer persistence of vision in a low light situation and help capture more reflections. Other types of waves and light sources are needed to broaden the applicability of our method.

Appendix A Per-pixel NDF

The integrals in Eqs. (5) and (6) for the single sine wave in Eq. (7) are

$$B = (B.x, B.z) = (a, b)K \tag{21}$$

$$M = (M.xx, M.xz, M.zz) = (a^2, ab, b^2)J \tag{22}$$

where

$$K = \frac{A(\sin(2(p_2 + ct)) - \sin(2(p_1 + ct))}{p_2 - p_1} \tag{23}$$

$$J = \frac{A^2(\sin(2(p_2 + ct)) - \sin(2(p_1 + ct)) + 2(p_2 - p_1)}{4(p_2 - p_1)} \tag{24}$$

To find p_1 and p_2 of each wave within the pixel, we project the pixel onto the wave plane, and then find the intersections of a line which is at the center of the projected square and in the direction of each wave, and the boundary of the projected square. $p(x, z)$ at the two intersections give p_1 and p_2 of the wave. Figure 19 illustrates our method.

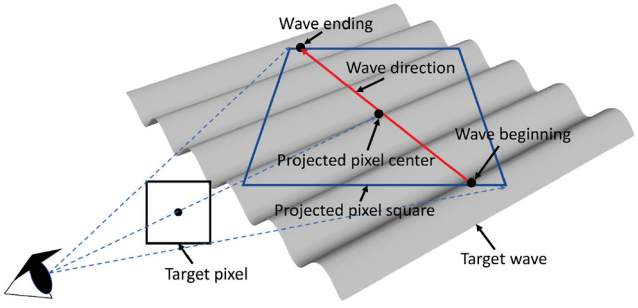


Fig. 19 Wave start and end in the image gives positions of p_1 and p_2 respectively.

Appendix B RDF approximation accuracy

We assume that the transformation from NDF to RDF is linear, which is incorrect and causes an error. We analyze this error by comparing our approximate RDF with the accurate RDF computed by sampling NDF. Figure 20 shows the relative error between them.

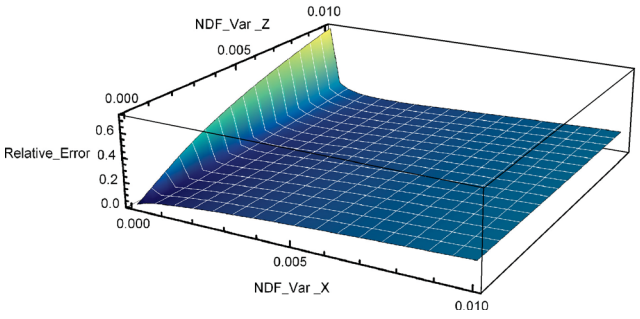


Fig. 20 Relative error of NDF to RDF transformation. Height and color show relative error while the other axis represents NDF variance in x and z axes. Blue to green to yellow represent the relative errors from low to high.

For each combination of x and z variances of the NDF, we average the error over multiple view directions, while aligning the view direction to the middle of the x and y axes. The relative error becomes larger when the x variance is low and z variance is high. The error also increases when both variances increase.

Appendix C Elliptical Gaussian approximation

As in Fig. 9, we place one circular kernel at the center of the elliptical Gaussian kernel while the others are placed in pairs along the major axis symmetrically. Thus the number of circular Gaussian kernels is $2N+1$, where N is an integer. We then have $3N+2$ parameters: weight w_a and variance σ_a^2 of the central kernel, N weights w_i and variances σ_i^2 of the symmetrical kernel pairs, and the distances d_i from the center to each symmetrical kernel pair.

Unlike Ref. [8], which used somewhat arbitrary simple formulae, we determine the parameters by minimizing the integral of the squared difference between the elliptical Gaussian kernel and our approximate multiple circular Gaussian kernels. Without loss of generality, we assume that the elliptical Gaussian's minor and major axes are on the x and y axis respectively. Then the error function ϵ is

$$\epsilon = \iint \left(\mathcal{G}_{\Sigma(\sigma_x,\sigma_y),(0,0)}(x,y) - \mathcal{C}(x,y)\right)^2 \mathrm{d}x\,\mathrm{d}y \tag{25}$$

$$\mathcal{C}(x,y) = w_a \mathcal{G}_{\Sigma(\sigma_a,\sigma_a),(0,0)}(x,y) + \sum_{i=1}^{N} \left(w_i(\mathcal{G}_{\Sigma(\sigma_i,\sigma_i),(0,d_i)}(x,y) + \mathcal{G}_{\Sigma(\sigma_i,\sigma_i),(0,-d_i)}(x,y))\right) \tag{26}$$

$$\Sigma(\sigma_x,\sigma_y) = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{27}$$

where $\mathcal{G}$ is a 2D Gaussian distribution function as in Eq. (2).

We can determine the optimal parameters by fixing σ_x to 1 since all other parameters can be scaled relative to σ_x. We used Mathematica to obtain an analytic formula for the double integral in Eq. (25) as a function of σ_y and the $3N+2$ parameters, for each of $N = 1, 2$, and 4. Here is an example for $N = 1$, with three circular Gaussian kernels:

$$\epsilon = \frac{\dfrac{w_a^2}{\sigma_a^2} + \dfrac{8w_a w_1 \mathrm{e}^{-\frac{d_1^2}{2\left(\sigma_a^2+\sigma_1^2\right)}}}{\sigma_a^2+\sigma_1^2} - \dfrac{4w_a}{\sqrt{\left(\sigma_a^2+1\right)\left(\sigma_a^2+\sigma_y^2\right)}}}{4\pi} + \frac{\dfrac{2b^2\left(\mathrm{e}^{-\frac{d_1^2}{\sigma_1^2}}+1\right)}{\sigma_1^2} - \dfrac{8w_1 \mathrm{e}^{-\frac{d_1^2}{2\left(\sigma_1^2+\sigma_y^2\right)}}}{\sqrt{\left(\sigma_1^2+1\right)\left(\sigma_1^2+\sigma_y^2\right)}} + \dfrac{1}{\sigma_y}}{4\pi} \tag{28}$$

We have similar but more complex formulae for five and nine circular Gaussian kernels. Figure 21 shows the optimization error.

Next, we sampled σ_y at regular intervals and computed the optimal parameters for each of the sampled values using Mathematica's `NMinimize` function. The parameters for an arbitrary value of σ_y are obtained by interpolation using the curve fitting function of Mathematica. Results of the curve fitting are given in Tables S1–S3 in the ESM.

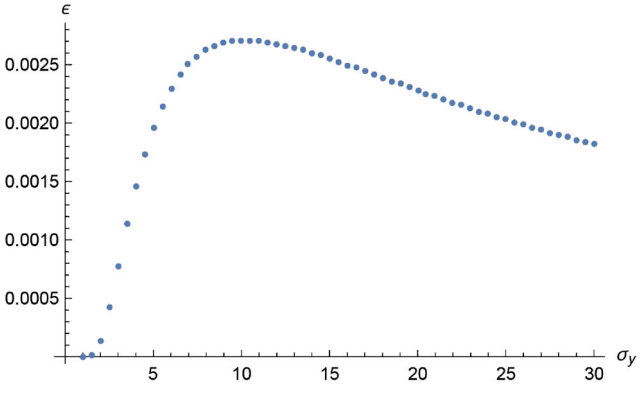


Fig. 21 Error ϵ (Eq. (25)) for three circular Gaussian kernels. The horizontal axis is the major axis standard deviation σ_y.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Nos. JP15H05924, JP18H03348, and JP20H05954.

Electronic Supplementary Material The animation result video and fitting parameter data are available in the online version of this article at `https://doi.org/10.1007/s41095-021-0204-1`.

References

[1] Gonzalez-Ochoa, C.; Holder, D.; Cook, E. Froma calm puddle to a stormy ocean: Rendering water in Uncharted. In: Proceedings of the ACM SIGGRAPH 2012 Talks, Article No. 3, 2012.

[2] Hopper, R.; Wolter, K. The water effects of pirates of the Caribbean: Dead men tell no tales. In: Proceedings of the ACM SIGGRAPH 2017 Talks, Article No. 31, 2017.

[3] Lottes, T. Fxaa. 2009. Available at `http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA$\backslash${\_}WhitePaper.pdf`.

[4] Korein, J.; Badler, N. Temporal anti-aliasing in computer generated animation. *ACM SIGGRAPH Computer Graphics* Vol. 17, No. 3, 377–388, 1983.

[5] Norton, A.; Rockwood, A. P.; Skolmoski, P. T. Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space. In: Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques, 1–8, 1982.

[6] Dupuy, J.; Heitz, E.; Iehl, J. C.; Poulin, P.; Neyret, F.; Ostromoukhov, V. Linear efficient antialiased displacement and reflectance mapping. *ACM Transactions on Graphics* Vol. 32, No. 6, Article No. 211, 2013.

[7] Bruneton, E.; Neyret, F.; Holzschuch, N. Real-time realistic ocean lighting using seamless transitions from geometry to BRDF. *Computer Graphics Forum* Vol. 29, No. 2, 487–496, 2010.

[8] McCormack, J.; Perry, R.; Farkas, K. I.; Jouppi, N. P. Feline: Fast elliptical lines for anisotropic texture mapping. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, 243–250, 1999.

[9] Cabral, B.; Olano, M.; Nemec, P. Reectionspace image based rendering. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, 165–170, 1999.

[10] McAuley, S.; Hill, S.; Martinez, A.; Villemin, R.; Pettineo, M.; Lazarov, D.; Neubelt, D.; Karis, B.; Hery, C.; Hoffman, N. et al. Physically based shading in theory and practice. In: Proceedings of the ACM SIGGRAPH 2013 Courses, Article No. 22, 2013.

[11] Heitz, E.; Dupuy, J.; Hill, S.; Neubelt, D. Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics* Vol. 35, No. 4, Article No. 41, 2016.

[12] Dupuy, J.; Heitz, E.; Belcour, L. A spherical cap preserving parameterization for spherical distributions. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 139, 2017.

[13] Shinya, M. Spatial anti-aliasing for animation sequences with spatio-temporal filtering. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 289–296, 1993.

[14] Nehab, D.; Sander, P. V.; Isidoro, J. R. The real-time reprojection cache. In: Proceedings of the ACM SIGGRAPH 2006 Sketches, 185, 2006.

[15] Scherzer, D.; Jeschke, S.; Wimmer, M. Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In: Proceedings of the 18th Eurographics Conference on Rendering Techniques, 45–50, 2007.

[16] Shannon, C. E. Communication in the presence of noise. *Proceedings of the IRE* Vol. 37, No. 1, 10–21, 1949.

[17] Grant, C. W. Integrated analytic spatial and temporal anti-aliasing for polyhedra in 4-space. *ACM SIGGRAPH Computer Graphics* Vol. 19, No. 3, 79–84, 1985.

[18] Zwicker, M.; Pfister, H.; van Baar, J.; Gross, M. EWA volume splatting. In: Proceedings of the Conference on Visualization, 29–36, 2001.

[19] Olano, M.; Baker, D. LEAN mapping. In: Proceedings of the ACM SIGGRAPH Symposiumon Interactive 3D Graphics and Games, 181–188, 2010.

[20] Becker, B. G.; Max, N. L. Smooth transitions between bump rendering algorithms. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 183–190, 1993.

[21] Wu, L. F.; Zhao, S.; Yan, L. Q.; Ramamoorthi, R. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. *ACM Transactions on Graphics* Vol. 38, No. 4, Article No. 137, 2019.

[22] Cook, R. L.; Porter, T.; Carpenter, L. Distributed ray tracing. In: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, 137–145, 1984.

[23] Smith, B. Geometrical shadowing of a random rough surface. *IEEE Transactions on Antennas and Propagation* Vol. 15, No. 5, 668–671, 1967.

[24] Ross, V.; Dion, D.; Potvin, G. Detailed analytical approach to the Gaussian surface bidirectional reectance distribution function specular component applied to the sea surface. *Journal of the Optical Society of America A* Vol. 22, No. 11, 2442–2453, 2005.

Namo Podee is currently a Ph.D. student at Hokkaido University. He received his B.S. and M.S. degrees from Chulalongkorn University and Hokkaido University, in 2013 and 2017, respectively.

Nelson Max is an emeritus Distinguished Professor at the University of California, Davis. He received his Ph.D. degree in mathematics from Harvard University in 1967. His research interests are in the areas of scientific visualization, computer animation, realistic computer graphics rendering, multi-view stereo reconstruction, and augmented reality.

Kei Iwasaki received his B.S., M.S., and Ph.D. degrees from the University of Tokyo, in 1999, 2001, and 2004, respectively. He is currently an associate professor at Wakayama University.

Yoshinori Dobashi is a professor at Hokkaido University, Japan. His research interests center on computer graphics, including realistic image synthesis, efficient rendering, and sound modeling for virtual reality applications. He received his B.E., M.E., and Ph.D. degrees in engineering in 1992, 1994, and 1997, respectively, from Hiroshima University. He worked at Hiroshima City University from 1997 to 2000 as a research associate.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this licence, visit `http://creativecommons.org/licenses/by/4.0/`.

Other papers from this open access journal are available free of charge from http://www.springer.com/journal/41095. To submit a manuscript, please go to https://www.editorialmanager.com/cvmj.