Adaptive Supersampling for Rendering Reflections on Water Surfaces

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

Abstract: Reflections on a dynamic surface such as a water surface are difficult render accurately in real-time due to aliasing from its highly reflective sub-pixel geometric structures. We propose a real-time rendering method for water surfaces that can remove both spatial and temporal aliasing while preserving the reflections from the sub-pixel geometric details. Our method first renders an aliasing-free image by approximating the distribution of normals due to the sub-pixel details with a Gaussian distribution. Then, our method detects the pixels that include significant high-contrast reflection detail from the sub-pixel geometry and renders sub-pixel reflections by supersampling.

Keywords: real-time rendering, anti-aliasing, reflection, water surface, adaptive supersampling

1. Introduction

Aliasing has been an important problem since the beginning of computer graphics. It is an artifact that is created when the data is not sampled enough. For example, if we only sample one position in a pixel that has two surfaces with different attributes inside, the result will most likely be unpredictable. This problem is even more prominent in water surface rendering.

Water surfaces usually have a high detail and animated geometric structure. This contributes to aliasing in both spatial and temporal domains. The traditional rendering method only samples the surface once per pixel, which is not enough most of the time for water surfaces. Many systems utilize the supersampling method and geometric simplification to tackle this problem. Supersampling is a method to sample a pixel multiple times to decrease aliasing. However, its computational cost is too high. Geometric simplification removes those details on the water surface that could cause any aliasing. This method is very cost-effective but it also removes the details from the rendering result.

We propose a method that utilizes both supersampling and geometric simplification to have a spatio-temporal anti-aliasing solution for rendering water surfaces comprised of sine waves. Our method is cost-effective yet provides detail and an accurate result.

Our main contributions are:

- A sine waves clamping method that removes waves causing aliasing and approximates their reflection contribution.
- A method that evaluates the reflection detail of each pixel.
- A method that computes parallel super sampling of each pixel proportionally to its evaluated detail.
- A demonstration that by combining our methods, we can render detailed anti-aliased reflections on water surfaces in real-time.

2. Related work

This research is built on our previous work [9], which can remove both temporal and spatial aliasing. However, it erases details from the reflection as well. Related work to preserve the details can be categorized into two groups: adaptive anti-aliasing and analytic anti-aliasing.

Adaptive anti-aliasing or supersampling is mainly about multi-sampling each pixel according to some evaluation. Lau and Rynson [4] introduce adaptive supersampling to remove hidden surfaces at the edge of geometry. Jin et al. [3] evaluate pixels in both image and object space then multi-sample them with ray tracing accordingly. Barringer et al. [1] render a scene first on a GPU and then evaluate and sample each pixel if needed with a CPU. Hollander et al. [2] apply adaptive supersampling in deferred rendering by using a supersampling g-buffer to help evaluate and shade each pixel appropriately. Lee et al. [5] introduce tile-based adaptive supersampling and undersampling ray tracing. Marrs et al. [6] remove artifacts from temporal anti-aliasing [10], [11] by evaluating artifact pixels and multi-sample them with ray tracing. These methods usually need a small number of additional samples (e.g. less than 16) per pixel. However, for rendering water surfaces, especially surfaces in a distance, the pixels can contain highly detailed geometries and more samples are usually required.

[1] Hokkaido University
[2] Wakayama University
[3] University of California, Davis
[a)] namo@ime.ist.hokudai.ac.jp

Analytic anti-aliasing is about coming up with a mathematical solution to solve or reduce the aliasing issue. We will borrow a lot of mathematical analyses from this type of research. Clamping [7] evaluates and removes any aliasing source from a texture by using the Nyquist theorem. EWA volume splatting [12] is a volume rendering anti-aliasing method by using elliptical Gaussian reconstruction kernels. LEAN mapping [8] approximates the accumulation of normal map values within one pixel as a Gaussian distribution. This helps remove aliasing from normal map sampling.

3. Proposed method

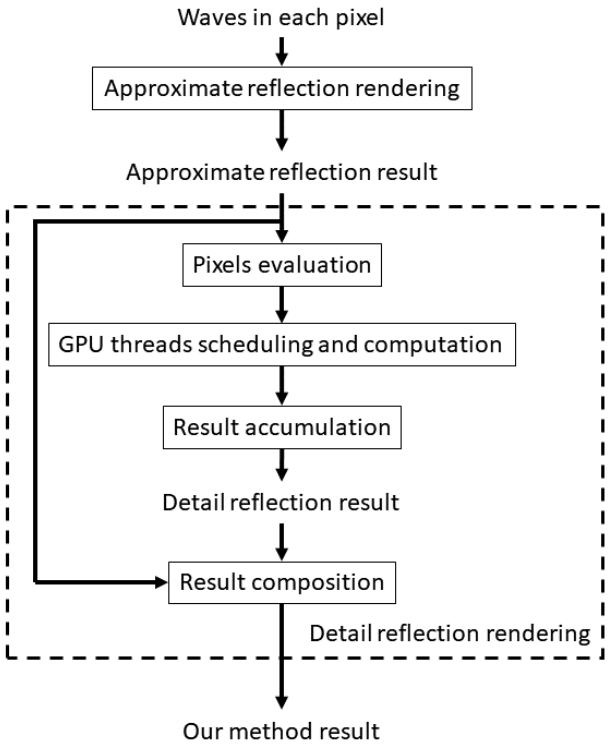


Fig. 1: The overview of our method.

Our main objective is to render the reflection of disk light sources on a water surface comprised of sine waves, without aliasing and without losing any details. An overview of our method will be explained first then we will go into the detail in each section.

Our method can be separated into two parts: an approximate reflection rendering and a detail reflection rendering. As shown in **Fig. 1**, our method starts with the approximated reflection rendering, which shares a lot of similarities to our previous work [9], to create a reflection result without aliasing and detail. Then, the detail reflection rendering evaluates the detail potential of each pixel of the approximated result. After that, it computes the multi-sampling of each pixel according to its evaluation. In the end, the multi-sampling result is combined and composed with the approximated result to create a final result.

3.1 Approximate reflection rendering

In our method, the water surface is represented by a combination of sine waves with different frequencies and amplitudes. We want to remove spatial-aliasing by removing every sine wave that causes spatial-aliasing from the water surface geometry, and then compensating for its removal by adding its contribution to the LEAN mapping reflection rendering.

First, we utilize the clamping method [7] to identify which sine waves will cause spatial-aliasing on each pixel. To explain further, each sine wave in a pixel will be projected to the screen space, then any sine wave whose projected frequency is larger than the Nyquist frequency is considered to be a spatial-aliasing wave. Every spatial-aliasing wave will be removed from the geometric calculation of the water surface.

Second, the clamped geometry will be rendered by our temporal ray tracing to remove any temporal-aliasing that it might have. Temporal ray tracing is done by calculating the time-integrated light contribution between the current frame and the previous frame. This can be done easily in our disk light source scenario. Our method calculates the intersection points between the plane of the disk light source and the reflected rays for both the current and the previous frames. Then we calculate the overlapped length between the disk light source and the line segment between the two intersection points. The time-integrated light contribution is calculated by the ratio of the length of the overlapped line segment to that of the line segment between the two intersection points.

Third, we have to compensate for the removed sine waves for any pixels where the sine waves causing spatial-aliasing. We combine the removed sine waves and approximate the normal distribution function (NDF) for those waves with a Gaussian distribution using the LEAN mapping method [8]. Then we transform the NDF to the reflection distribution function (RDF) by using Gaussian kernel transformation technique in EWA volume splatting [12]. Then the light contribution can be calculated from a convolution of the RDF and the disk light source. We calculate this convolution by using numerical integration on a GPU.

Lastly, we can combine the result of temporal ray tracing and RDF-disk-light convolution to have a final approximated result. However, the result of the two methods is too different. Combining them directly will lead to a non-smooth result. Thus, we transition the wave clamping by a gradually reducing the amplitude of each wave when its projected frequency gets close to Nyquist frequency. With this method, we can blend the result between the two methods very smoothly.

3.2 Detail reflection rendering

The approximated result that we created earlier doesn't contain enough details in the reflection because the spatial-aliasing waves are approximated with a Gaussian distribution NDF, which is not accurate enough. An accurate result can be obtained by supersampling each pixel. However, wa-

ter surfaces are very complicated and some pixels might need even more than 1000 samples, which is impossible in real time for all pixels with a traditional method. Our method tackles this problem by efficiently scheduling a million GPU threads to help compute a few pixels that make a difference in the detail of the overall reflection result.

First, our method evaluates the importance value of each pixel by using its pixel value from the approximated result. In this current stage, we simply put the approximated result into a Gaussian function, so that a pixel on the edge of the light source reflection area is considered more important, and any pixel that is not in that reflection at all is ignored. This can be described as:

$$v_i = g_\rho(R_a(i) - c), \tag{1}$$

where v_i is the importance value of i-th pixel, g_ρ is a 1D Gaussian function with variance of ρ^2, $R_a(i)$ is the approximated result of pixel i, and c is a center of the Gaussian function. We adjust both ρ and c manually for each scene. Any pixel that has an importance value above a threshold will generate a sampling task, which contains its pixel position, sampling position, and the importance value. Every sampling task will be added to a sampling task list.

Second, our method calculates a prefix sum of the importance value of all tasks in the sampling task list. This can be done in parallel on a GPU as well.

Third, our method dispatches a fixed number of GPU threads to work on sampling tasks. Our objective is to assign a task to a group of threads, whose size is as proportional as possible to the task's importance value. This can be done by associating each GPU thread id to each task's prefix sum value of importance value. Each thread first calculates the following searching value:

$$s_k = k \sum_{j=0}^{N} v_j, \tag{2}$$

where k is a GPU thread id, N is the size of the sampling tasks, v_j is the importance value of j-th task. Then we use a binary search to find a task that has the maximum prefix sum value that is still lower or equal to its s_k value. Now each GPU thread knows which task to execute.

Fourth, each GPU thread computes its task by randomly sampling the position within the task's pixel then using our temporal ray tracing to shade that ray.

Fifth, the result from each GPU thread is atomically collected into the detail result.

Last, the detail result is blended with the approximated result by using a weighting value for the detail result. We can calculate the weight a of i-th pixel of the detail result from the ratio of a size of threads that working on this pixel and the sample size needed to satisfy the Nyquist limit, which is:

$$a_i = clamp(M * \frac{v_i}{\sum_{j=0}^{N} v_j} * \frac{f_{nyquist}}{max(f_k)}, 0, 1), \tag{3}$$

where M is a total thread size, v_i is the importance value of i-th pixel, f_k is a projected frequency of k-th sine wave, and $f_{nyquist}$ is the Nyquist frequency. The blended result R of i-th pixel will be:

$$R_i = (1 - a_i) * R_{a,i} + a_i * R_{d,i}, \tag{4}$$

where $R_{a,i}$ is the result of the approximate reflection rendering, and $R_{d,i}$ is that of the detail reflection rendering.

4. Results

We implemented our method using OpenGL and our method runs entirely on a GPU. We currently experiment with just one scene, which is a sunset scene, with parameters: $c = 0.1, \rho = 0.04$. **Fig. 2** shows the comparison result between the reference image, the result without anti-aliasing, the result with 32x MSAA, and the result with our method. The reference image is calculated by the tiling 128x supersampling method. **Table 1** shows the computation time comparison of the mentioned methods. **Fig. 3** shows the internal result of our method. **Fig. 3a** and **Fig. 3b** show the result from the approximate and the detail reflection rendering method respectively. **Fig. 3c** shows the importance value of each pixel.

Table 1: The computational time comparison.

Rendering Method	Rendering Time
Reference	8326 ms
No AA	3 ms
32x MSAA	10 ms
Our Method	15 ms

Our method can achieve a similar result and detail to the reference image while being in real-time, unlike the 128x supersampling method.

The rendering times are measured on a laptop with an Intel Core i7 @ 2.50Ghz, Memory 16 GB, and an NVIDIA GeForce GTX 860M.

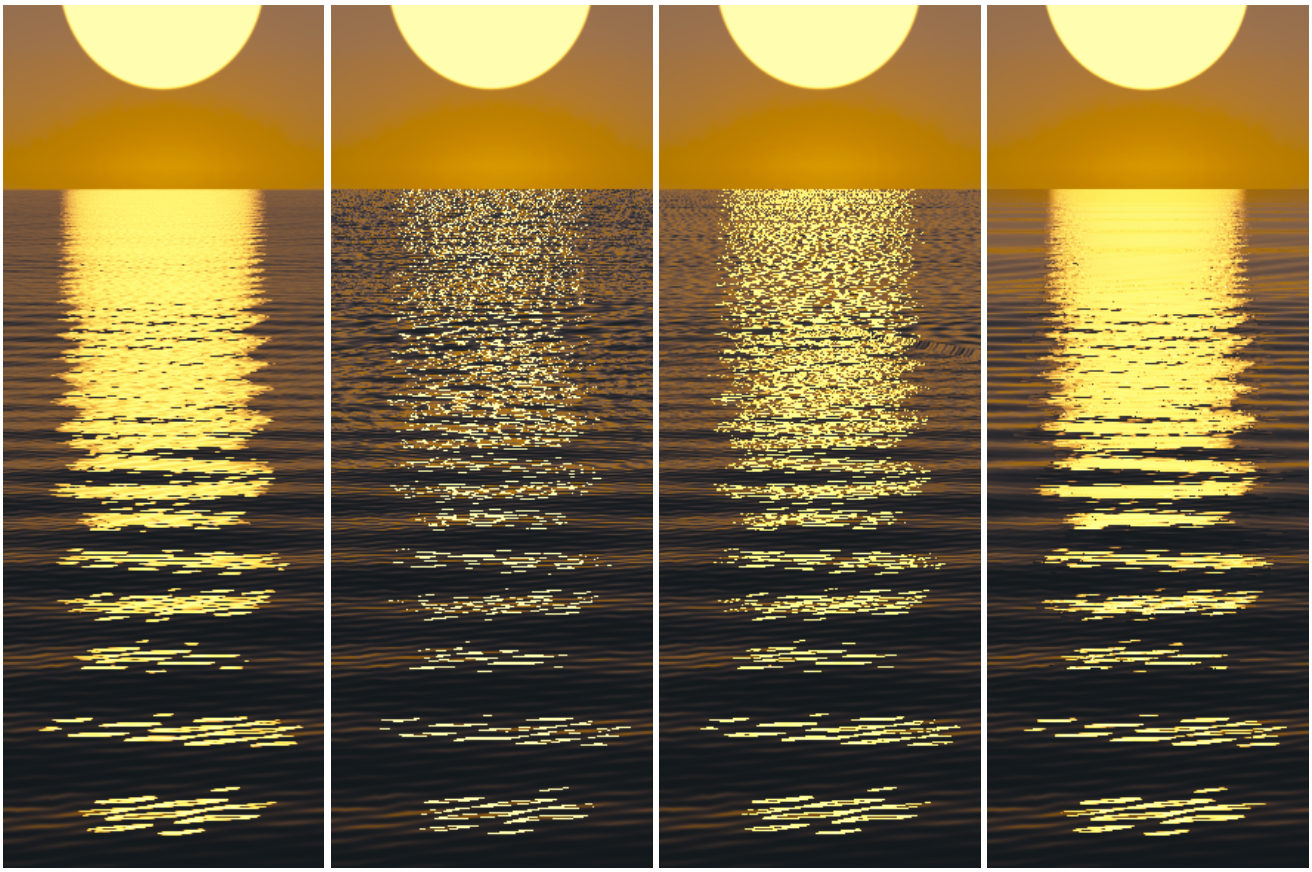
5. Conclusion

We have presented a real-time rendering method for moving water surfaces with a highly detailed reflection of a disk light source. However, many problems remain, namely: pixel importance value evaluation, limited geometry, and light source support. In our current progress, our method parameters need to be adjusted manually for each scene to achieve the best result. Also, our method only supports sine waves for the water surfaces geometry and disk light source for a light source type. These problems make our method still impractical for many real applications.

We are working on fixing these problems and making our research useful in rendering a beautiful reflections on a water surface in real-time.

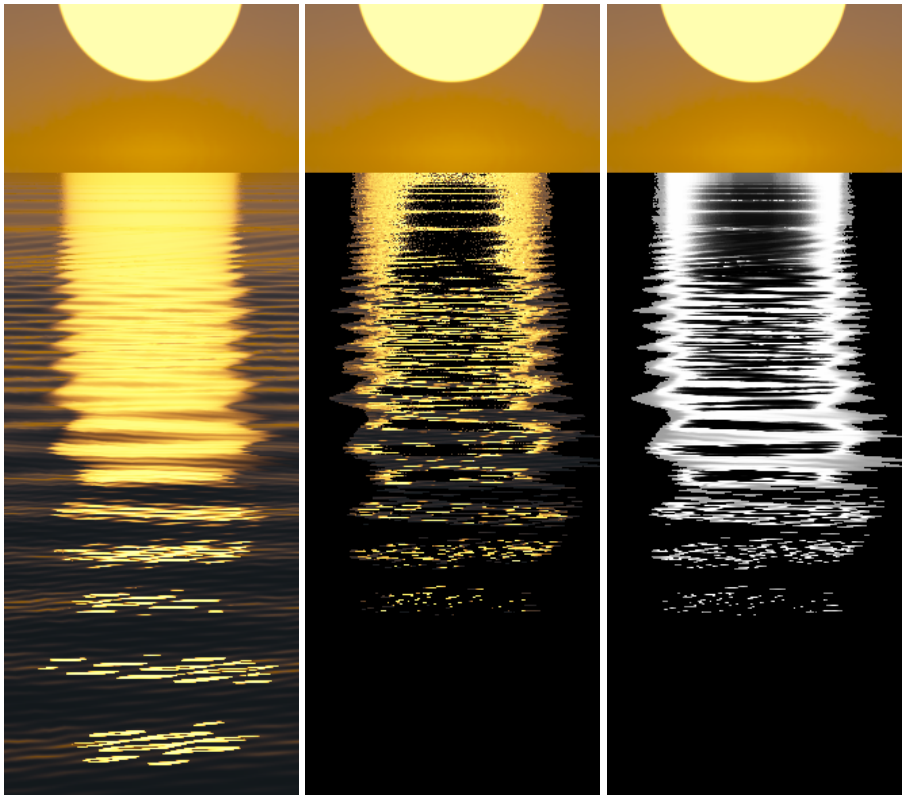
Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP18H03348.



(a) Reference image (b) Without AA (c) 32x MSAA (d) Our method

Fig. 2: The comparison result from the sunset scene.



(a) Approximate result (b) Detail result (c) Importance value

Fig. 3: The internal result of our method.

References

[1] Barringer, R. and Akenine-Möller, T.: A4: Asynchronous Adaptive Anti-Aliasing Using Shared Memory, *ACM Trans. Graph.*, Vol. 32, No. 4 (online), DOI: 10.1145/2461912.2462015 (2013).

[2] Holländer, M., Boubekeur, T. and Eisemann, E.: Adaptive Supersampling for Deferred Anti-Aliasing, *Journal of Computer Graphics Techniques (JCGT)*, Vol. 2, No. 1, pp. 1–14 (online), available from ⟨http://jcgt.org/published/0002/01/01/⟩ (2013).

[3] Jin, B., Ihm, I., Chang, B., Park, C., Lee, W. and Jung, S.: Selective and Adaptive Supersampling for Real-Time Ray Tracing, *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, New York, NY, USA, Association for Computing Machinery, p. 117–125 (online), DOI: 10.1145/1572769.1572788 (2009).

[4] Lau, R. W. H.: An Adaptive Supersampling Method, *Proceedings of the Third International Computer Science Conference on Image Analysis Applications and Computer Graphics*, ICSC '95, Berlin, Heidelberg, Springer-Verlag, p. 205–214 (1995).

[5] Lee, W.-J., Hwang, S. J., Shin, Y., Ryu, S. and Ihm, I.: Adaptive Multi-Rate Ray Sampling on Mobile Ray Tracing GPU, *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*, SA '16, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/2999508.2999523 (2016).

[6] Marrs, A., Spjut, J., Gruen, H., Sathe, R. and McGuire, M.: Adaptive Temporal Antialiasing, *Proceedings of the Conference on High-Performance Graphics*, HPG '18, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/3231578.3231579 (2018).

[7] Norton, A., Rockwood, A. P. and Skolmoski, P. T.: Clamping: A Method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space, *Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '82, New York, NY, USA, ACM, pp. 1–8 (online), DOI: 10.1145/800064.801252 (1982).

[8] Olano, M. and Baker, D.: LEAN Mapping, *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, New York, NY, USA, ACM, pp. 181–188 (online), DOI: 10.1145/1730804.1730834 (2010).

[9] Podee, N., Max, N., Iwasaki, K. and Dobashi, Y.: Temporal and Spatial Anti-Aliasing for Rendering Reflection on a Water Surface, *ACM SIGGRAPH 2019 Posters*, SIGGRAPH 2019, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/3306214.3338599 (2019).

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

[11] Tatarchuk, N., Karis, B., Drobot, M., Schulz, N., Charles, J. and Mader, T.: Advances in Real-Time Rendering in Games, Part I (Full Text Not Available), *ACM SIGGRAPH 2014 Courses*, SIGGRAPH '14, New York, NY, USA, Association for Computing Machinery, (online), DOI: 10.1145/2614028.2615455 (2014).

[12] Zwicker, M., Pfister, H., van Baar, J. and Gross, M.: EWA Volume Splatting, *Proceedings of the Conference on Visualization '01*, VIS '01, Washington, DC, USA, IEEE Computer Society, pp. 29–36 (online), available from ⟨http://dl.acm.org/citation.cfm?id=601671.601674⟩ (2001).