Fast orientable aperiodic ocean synthesis using tiling and blending

NICOLAS LUTZ, University of Sherbrooke, Canada
ARNAUD SCHOENTGEN, Ubisoft, Canada
GUILLAUME GILET, University of Sherbrooke, Canada

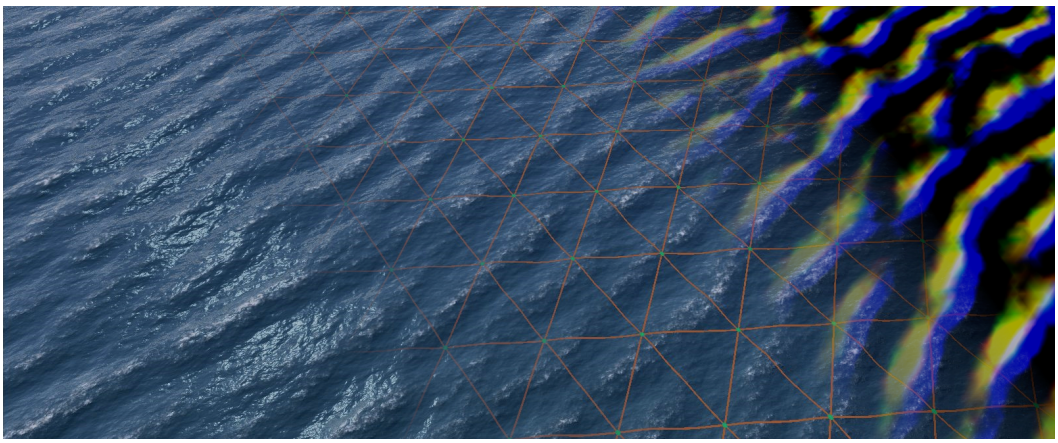

Fig. 1. We propose to apply the by example texture synthesis method of Heitz and Neyret [2018], tiling and blending, to aperiodically tile and filter the displacements and normals computed by the periodic ocean simulation algorithm of Tessendorf [2001]. Left: aperiodic deep ocean rendering; middle: tiling vertices visualized; right: displacement visualized.

The simulation and rendering of the surface of a deep ocean are typically carried by computing a mesh displacement through an Inverse Fast Fourier Transform (IFFT) of an animated ocean spectrum. This process generates a spatially periodic ocean displacement that can be tiled to pave a large ocean surface. However, this creates tiling artifacts for large oceans. This effect can be toned down by mixing the displacement with noise, which disturbs the appearance of the ocean, or by overlapping the result of several IFFT at different scales, which increases computation times, all while not fully removing the periodic aspect. We propose to instead use tiling and blending, a procedural generation algorithm popular for real-time texture synthesis, in order to generate variations of the mesh displacement. This method also enables us to author the direction of the waves using a flow map. We show that this method is especially fast and can create a fully aperiodic ocean with minimal downsides.

CCS Concepts: • **Computing methodologies** → **Procedural animation**; *Rendering*; *Texturing*; Mesh models.

Authors' addresses: Nicolas Lutz, University of Sherbrooke, 2500 Bd de l'Université, Sherbrooke, Canada, nicolas.lutz@usherbrooke.ca; Arnaud Schoentgen, Ubisoft, 5505 Bd Saint-Laurent, Montréal, Canada, arnaud.schoentgen@ubisoft.com; Guillaume Gilet, University of Sherbrooke, 2500 Bd de l'Université, Sherbrooke, Canada, guillaume.gilet@usherbrooke.ca.

ACM 2577-6193/2024/7-ART
https://doi.org/10.1145/3675388

Additional Key Words and Phrases: Ocean simulation, procedural generation, texture synthesis, animated textures

ACM Reference Format:
Nicolas Lutz, Arnaud Schoentgen, and Guillaume Gilet. 2024. Fast orientable aperiodic ocean synthesis using tiling and blending. *Proc. ACM Comput. Graph. Interact. Tech.* 7, 3 (July 2024), 22 pages. https://doi.org/10.1145/3675388

1 INTRODUCTION

The simulation and rendering of ocean surfaces are mature topics. They have been used to populate virtual environments with realistic oceans, from movies to video games. In this work, we are interested in the high-performance generation of large-scale oceans, targeting real-time applications such as video games. In such applications, deep ocean waves are usually modeled as an animated displacement applied at run-time onto a tessellated plane geometry. In this context, such models are subject to several hard constraints: they should be computationally efficient, be able to represent a wide range of ocean phenomena over an unbounded size, possess high-quality filtering schemes to avoid aliasing, seem aperiodic to avoid tiling artifacts, and enable artistic control to author the appearance of the ocean.

A common model, widely used in production, was proposed by Tessendorf [2001]. It computes wave displacements from an ocean spectrum taken from real oceanographic analysis and a time-dependent phase using an Inverse Fast Fourier Transform (IFFT). In its classic form, the Tessendorf model yields a periodic displacement that is tiled on the entire surface. Many improvements have been proposed by the community over the past 20 years, thus satisfying most constraints for real-time applications. However, the periodic aspect of generated oceans still remains an issue.

In the literature, there are are two main solutions to create variety on the ocean surface: the first approach consists in blending the surface of the ocean with an aperiodic noise function, such as Perlin noise [NVIDIA 2011; Rydahl 2009]. The second consists in subdividing the spectrum into several sub-spectra, computing their IFFT at different mesh scales and adding the resulting sub-displacement maps together (see Figure 2) [Dupuy and Bruneton 2012]. As we show in Section 3.1, neither approach fully hides their innate periodicity.

In this work, we propose an alternative to hiding the periodicity of ocean waves by fully removing it using a real-time by example texture synthesis algorithm. Our approach is presented in Section 4; it consists in synthesizing an ocean from an input periodic displacement map and normal map generated with the ocean generation model of Tessendorf [2001], using the tiling and blending algorithm of Heitz and Neyret [2018]. Our method meets the requirements of real-time applications (efficiency, quality, genericity and aperiodicity). In particular, it allows to generate a fully aperiodic ocean, contrary to previous models. It is also simple enough to be easily integrated in most applications. Figure 2 (right) illustrates our approach to compute a final ocean displacement map compared to the computation of several sub-displacements.

In Section 5, we show that our approach can be simply extended to apply an arbitrary flow map that locally controls the direction of the waves for artistic authoring purposes. In Section 6, we demonstrate that our approach is compatible with LEAN mapping filtering [Olano and Baker 2010] for rendering a credible specular lobe on the ocean surface at any view distance, with or without using a flow map.

In Section 7, we describe our results in term of quality and performance: we show that our approach is faster to compute than combining multiple sub-displacements. We also show that our approach better preserves the desired appearance of the ocean while better hiding tiling artifacts as shown by the study of the autocorrelation function of our ocean displacement. Finally, we discuss the limits of our approach in Section 8.

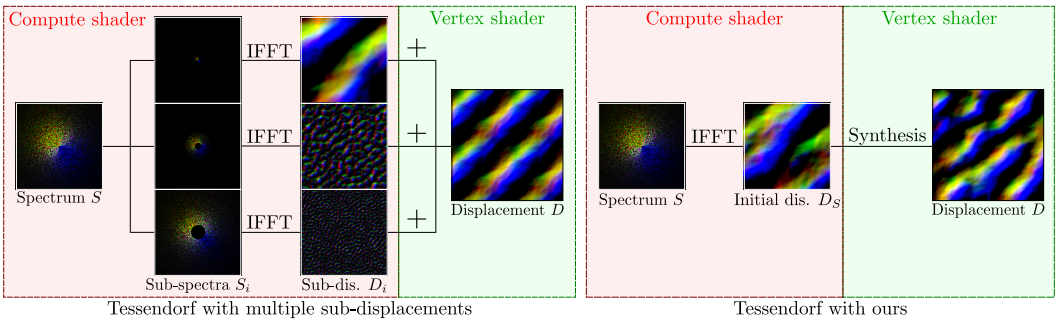

Fig. 2. Overview of our technique compared to the method of Dupuy and Bruneton [2012]. The generation of an aperiodic-looking ocean is typically carried by subdividing a spectrum into multiple sub-spectra representing different frequencies and summing the result of the IFFT of each sub-spectrum. Instead, we propose to compute a displacement map by synthesizing the result of a single IFFT on the whole spectrum in order to remove any periodicity and lower computation time.

2 RELATED WORK

Our work expands on real-time ocean simulation using an algorithm initially designed for real-time by-example texture synthesis. As such, in this section, we discuss previous works of both fields separately.

2.1 Real-time ocean simulation

In computer graphics, ocean simulation is usually carried from the work of Tessendorf [2001] using Inverse Fast Fourier Transform (IFFT) to compute and render a time-varying displacement map on a tessellated plane geometry. It allows for a simulation of ocean waves whose wave spectrum matches the statistics of a real ocean [Fréchot 2006; Horvath 2015; Lee et al. 2007]. Mitchell [2005] showed how to compute this model in real-time using GPU hardware. To improve the realism of the simulation, several improvements compatible with real-time applications have been proposed. Dupuy and Bruneton [2012] showed how to render and filter ocean whitecaps for agitated oceans. Horvath [2015] describes the practical application of several empirically-based, directional ocean wave spectra and introduces a swell parameter to control the elongation of the waves during the simulation.

Periodicity removal. The issue with computing a displacement map through an IFFT using the Tessendorf model [Tessendorf 2001] is that it results in a periodic displacement map, and therefore a periodic ocean surface. This periodicity is detrimental to the plausibility of the simulation as it creates visual artifacts. Breaking periodicity can be carried by mixing random noise with the periodic displacement map [NVIDIA 2011; Rydahl 2009]. However, mixing too little noise may fail to mask the periodicity, while mixing too much may disturb the appearance of the ocean.

Dupuy and Bruneton [2012] propose to compute several periodic sub-displacements from several IFFTs computed at different ranges of frequencies, and overlap them with different tiling scales, bringing the periodicity to the least common multiple of their scales. Bridson [2015] have also suggested to superimpose two periodic sub-displacements, and suggested to use a ratio between both tiling scales corresponding to an irrational number. In any case, the amplitude and tiling scales of each sub-displacement may be tweaked to author the look of the final ocean. However, periodic sub-displacements are still visible, especially those made of low frequency waves.

Our approach is compatible with any model that computes periodic oceans, and enables to efficiently and completely eliminate ocean periodicity while keeping its overall appearance.

Wave orientation control. Wave spectra depend on the wind properties such as wind direction and speed. In practice, the wind direction can be changed to orient the ocean waves. By default, paving an ocean with a spatially periodic ocean tile results in waves oriented along the same direction which is often too limited from an artistic control standpoint. Olano and Baker [2010] modeled an ocean that moves towards the coast by blending 4 different oceans with orthogonal waves whose weights vary according to the geometry of the coast. We instead propose a method in Section 5 exploiting our synthesis model which does not require the simulation of several oceans, sparing memory consumption and computation time.

Ocean filtering. Filtering both the geometry and appearance of the ocean is required to avoid aliasing artifacts. Filtering the geometry is often carried using a tessellation scheme of the mesh [Chiu and Chang 2006]. LEAN mapping [Olano and Baker 2010] may be used to accurately filter the ocean's normals for rendering specular highlights. LEADR mapping [Dupuy et al. 2013] can extend LEAN mapping by taking into account the masking-shadowing effect of displaced waves. Later, Grenier et al. [2022] showed a variance and covariance estimator that allowed them to LEAN map a surface using tiling and blending. We show in Section 6 that LEAN mapping approaches are compatible with our ocean tiling and blending approach.

2.2 Real-time texture synthesis

Real-time by-example texture synthesis consists in generating a texture using an exemplar as a parameter, meant to guide the aspect of the output. In the past, it was often carried using procedural noise algorithms [Lagae et al. 2010] such as Gabor noise [Galerne et al. 2012], random phase noise [Galerne et al. 2011; Gilet et al. 2014], or texton noise [Galerne et al. 2017]; or by using aperiodic tiling algorithms, such as Wang tiles [Cohen et al. 2003] or content exchange [Vanhoey et al. 2013]. Recently, Heitz and Neyret [2018] proposed an approach balanced between both algorithm families called tiling and blending. Tiling and blending is a fast, trivially filterable synthesis algorithm which consists in blending three hexagonal tilings on a regular grid with a variance- or histogram-preserving algorithm. Other tiling shapes may also be used for tiling and blending, which can lead to increased computational speeds when the number of tilings is two [Lutz et al. 2023].

To our knowledge, tools for texture synthesis have rarely been used for synthesizing an ocean. A notable instance is in the work of Vámošová [2014], who used Wang tiling to synthesize the displacement map of an ocean from a set of Wang tile exemplars. Wang tiling however offers limited variety, increases memory consumption and requires the computation of a set of tiles each frame. The IFFT was also often used in texture synthesis in the past, notably with the random phase noise model [Galerne et al. 2011].

In this work, we introduce a novel pipeline which can include any by-example synthesis algorithm to synthesize the displacement map of an ocean mesh in a vertex shader, and to synthesize its appearance in a fragment shader. We choose to showcase tiling and blending in particular due to its simplicity and overall trade-off between speed and quality.

3 BACKGROUND

In this section, we provide background related to key concepts and algorithms that are exploited throughout this paper for deep water simulation and the tiling and blending texture synthesis algorithm.

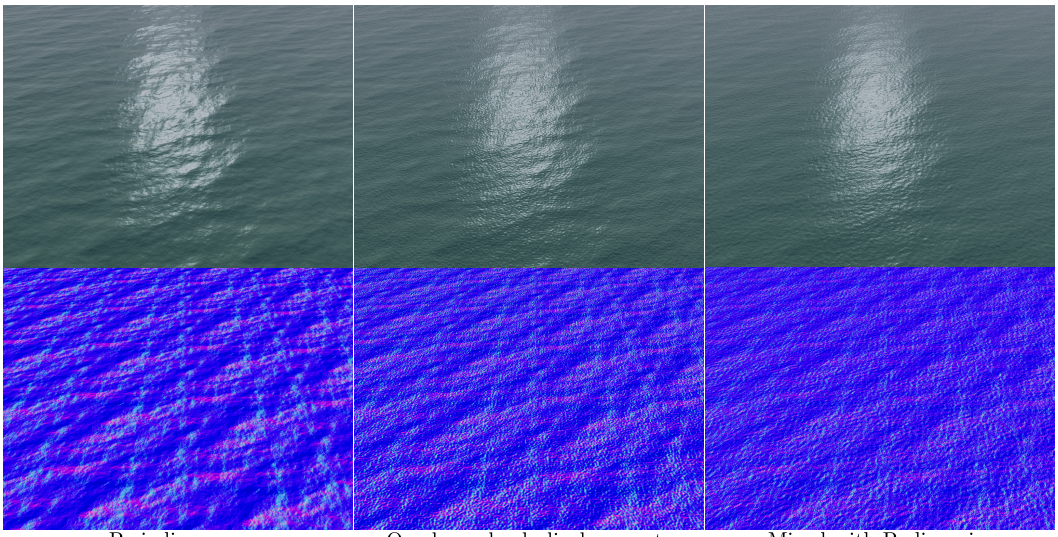

Fig. 3. Periodic ocean simulation (left), along with two different ways of hiding periodicity: combining several sub-displacements computed from sub-spectra (middle), and mixing a periodic ocean with Perlin noise. Top row shows a rendering of the ocean surface; bottom row shows the normal map with the vertical component scaled down for visualization purpose. Both methods still exhibit visible low frequency/high amplitude waves, while disturbing the original appearance. Note that these oceans have been generated using the Pierson-Moskowitz spectrum [Pierson Jr. and Moskowitz 1964].

3.1 Deep water simulation

The simulation of deep ocean waves is typically done by displacing a plane geometry using a space-time varying displacement map obtained from oceanographic wave profiles matching the look of a real ocean [Darles et al. 2011]. We define the displacement map of an ocean from a plane mesh as a function of position and time $D : X, \mathbb{R} \to \mathbb{R}^3$, where $X = \mathbb{R}^2$ is the continuous index set of the plane mesh. Without displacement, we consider that the object space coordinates are the projection of $\mathbf{x}$ onto a plane mesh, with a height of zero for any index $\mathbf{x} \in X$. Following this, the displaced position of a point $\mathbf{q}$ on the ocean from its initial position $\mathbf{p}$ is simply expressed through a position $\mathbf{x}$ and a time t as

$$\mathbf{q}(\mathbf{x}, t) = \mathbf{p} + D(\mathbf{x}, t). \tag{1}$$

In the Tessendorf model [Tessendorf 2001], the displacement map D is generated by computing a time-varying periodic displacement map through the IFFT of a time-varying ocean wave spectrum, usually in a compute shader, and repeating it to pave an ocean surface. We provide more details on the generation of D in appendix B. An additional normal map $N : X, \mathbb{R} \to \mathbb{R}^3$ encoding the analytical expression of the surface normals for any index of the plane mesh can also be generated to compute the shading of the displaced ocean. The normal map N can be computed from the surface slope, which itself can be analytically calculated using additional IFFTs [Tessendorf 2001].

Since both the displacement map D and the normal map N are periodic, this creates unnatural tiling artifacts on the water surface, visible in the left view of Figure 3. We explore two different propositions that have been used in the past to hide the periodicity of the ocean.

Hiding periodicity using random noise. Rydahl [2009] propose to remove periodicity by mixing the ocean with random noise. The implementation of NVIDIA [2011] in particular consists in

blending an initial periodic displacement map D_S computed from the IFFT of a spectrum S with a displacement map D_P generated with Perlin noise. In that case, a final displacement map D is computed as a linear interpolation of both maps:

$$D(\mathbf{x}, t) = (1 - w_P)D_S(\mathbf{x}, t) + w_P D_P(\mathbf{x}, t). \tag{2}$$

where w_P is the user-defined weight of the Perlin noise. We note that if D_P is spatially-stationary (i.e. its statistics are independent of $\mathbf{x}$, such as the mean), then $D(\mathbf{x}, t)$ is spatially-cyclostationary [Lutz et al. 2021] (i.e. its statistics are periodic on $\mathbf{x}$) with the periods of D_S, because it would then be the result of a linear combination between a periodic signal and a stationary signal.

In our experiments, we found that it is difficult to find an adequate balance by tweaking w_P: a small weight makes the periodicity too noticeable, a weight too large looks too much like Perlin noise, and any value in between carries both disadvantages such as in Figure 3.

Hiding periodicity using multiple sub-displacements. Dupuy and Bruneton [2012] instead propose to modify the approach of Tessendorf such that the spectrum is divided into n sub-spectra S_i for $0 \leq i \leq n-1$, which in turn yield n periodic sub-displacements D_i. Each sub-spectrum represents a user-defined range of frequencies of the original spectrum S. In that case, a final displacement map D is computed as

$$D(\mathbf{x}, t) = \sum_{i=0}^{n-1} D_i(\mathbf{x}, t). \tag{3}$$

Each sub-displacement D_i may be independently edited to control the aspect of the ocean. This includes controlling the frequency band corresponding to each sub-spectrum S_i, the scaling factor applied to the resulting displacement D_i, or the size of the tile corresponding to a period of the sub-displacement map.

Using different sizes for each sub-displacement map does not remove the periodicity but instead increases it to the least common multiple of the periods of the different sub-displacements. Since the sub-displacement maps remains periodic, repeating patterns may still be visible.

In our experiments, we found that the periodicity remains especially noticeable for low frequency components of the displacement map as shown in Figure 3. We also note that this technique increases the amount of IFFTs to compute, increasing both memory consumption and computation times.

The sub-displacements D_i must be computed for each step of time, typically on the GPU. The derivatives of each sub-displacement D_i are also computed to generate the normals, used for rendering the water surface. Optionally, the Jacobian of the displacement can also be calculated to render ocean whitecaps [Dupuy and Bruneton 2012].

In our work, we use different oceanographic wave spectra with the Tessendorf model, including Phillips [Tessendorf 2001], JONSWAP [Horvath 2015], and Pierson-Moskowitz [Pierson Jr. and Moskowitz 1964]. Note that other wave spectra are straightforwardly compatible with the ocean synthesis model we present in Section 4.

3.2 Tiling and blending

Tiling and blending was introduced by Heitz and Neyret [2018] as a fast way to synthesize an output texture from an exemplar texture $E : X \longrightarrow S$ in the fragment shader, where X is an index space and S is a value space. In its original form, it works by blending 3 regular hexagonal tilings such that each texel is overlapped by exactly 3 different tiles which we refer to as "synthesis tiles". Blending weights are maximal at the center of the synthesis tiles, and decrease down to 0 at the edges. It has been generalized by Lutz et al. for an arbitrary number of synthesis tiles of arbitrary

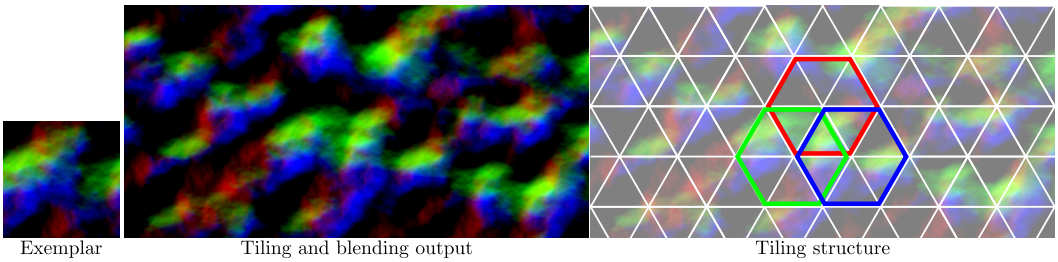

Fig. 4. Tiling and blending synthesis, proposed by Heitz and Neyret [2018]. It enables to aperiodically tile a surface by rearranging and blending hexagonal tiles, whose content is taken randomly in an exemplar texture. We use this synthesis algorithm to aperiodically tile a displacement map of our ocean, used as the exemplar of this Figure. The exemplar is a displacement map generated by computing the IFFT of a Phillips spectrum [Tessendorf 2001].

shapes [Lutz et al. 2023] as the following equation, in the stationary case:

$$I(\mathbf{x}) = \sum_{i=0}^{n-1} w_i\left(\mathbf{x}\right)\left(E_i\left(\mathbf{x}\right) - \mu\right) \ + \mu, \tag{4}$$

where E_i is the content of a tiling, defined by

$$E_i(\mathbf{x}) = E\left(\mathbf{x} + \mathbf{h}\left(k_i\left(\mathbf{x}\right)\right)\right). \tag{5}$$

In Equation 4, n is the number of overlapping tilings, $\{k_i\}$ are tiling functions which yield a unique index for each synthesis tile; $\{w_i\}$ are blending weights, subjected to $\sum_{i=0}^{n-1} w_i^2 = 1$; $\mathbf{h}$ is a random uniform sampler which yields a random offset from a tile index used as a seed; finally, μ is the spatial mean of E, which can be estimated as an average of the values of E. The normalization of the squared weights and the shift by the spatial mean are used to simultaneously preserve both the spatial mean and the spatial variance of the exemplar E into the output I, as shown by Heitz and Neyret [2018]. Tiling and blending can be pre-filtered by MIP-mapping the exemplar E itself, and using Equation 4 on the appropriate resolution at run time.

In the hexagonal tiling and blending of Heitz and Neyret [2018], tilings take the form of 3 overlapping hexagonal tilings k_0, k_1 and k_2 such that their overlap forms a triangular grid made of all hexagon vertices, as in Figure 4. This makes it possible to sample offsets from each triangle vertex, and compute weights according to barycentric coordinates in each triangle, as in the implementation of Deliot and Heitz [2018].

To make the synthesis more authorable, Burley [2019] propose to control the blending weights with an exponent to tweak the appearance of the output texture, where higher exponents transition faster from one synthesis tile to another, avoiding ghosting artifacts from blendings but creating discontinuities if the exponent is to high. Lutz et al. [2021] also showed that the autocorrelation function is not well-preserved by tiling and blending; we show that this property is interesting for generating a more realistic ocean in Section 7.

We propose to combine the tiling and blending of this section with the model of Tessendorf mentioned in Section 3.1 to compute an aperiodic ocean.

4 OCEAN SYNTHESIS MODEL

In this section, we present our ocean synthesis model through the by-example synthesis of time-dependent displacement maps and normal maps.

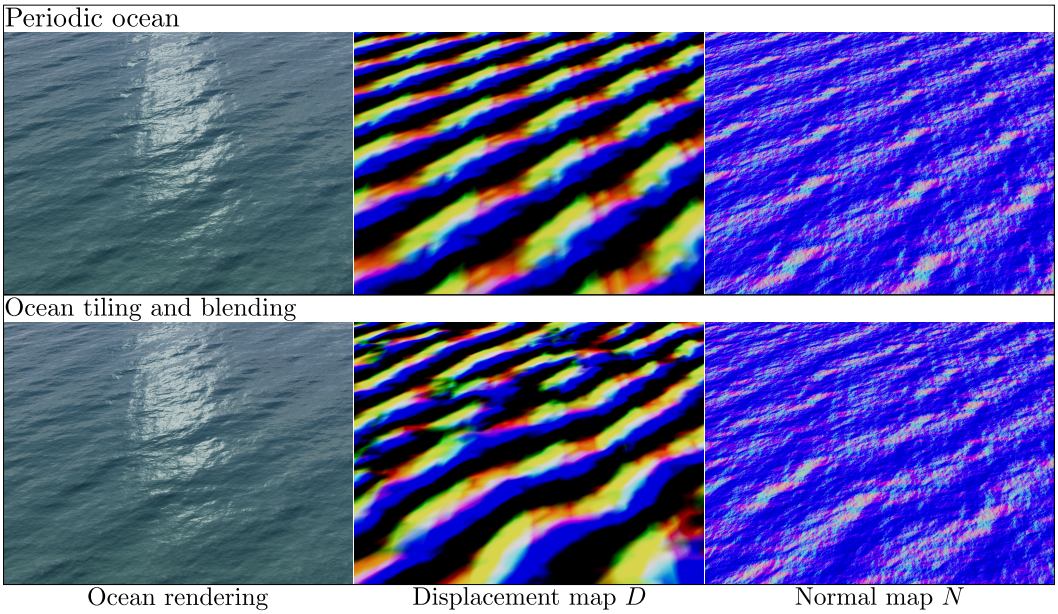

Fig. 5. Results of our aperiodic ocean synthesized with tiling and blending, with a periodic ocean as reference on top: for our final rendering, a visualization of the synthesized displacement map D, and a visualization of the synthesized normal map N (with the z-axis scaled down for visualization).

4.1 Overview

Our model consists in using tiling and blending to synthesize an ocean from a single initial displacement map D_S computed using the Tessendorf model [Tessendorf 2001] at each frame, as well as on its corresponding initial normal map N_S. The proposed approach essentially boils down to using the tiling and blending described in Section 3.2 on D_S in the vertex shader, and using the resulting displacement map to displace the ocean mesh. Normals are then computed for shading using the same synthesis model in the fragment shader. We show the result of using tiling and blending on a periodic ocean in Figure 5.

4.2 Displacement map

In our model, an initial displacement map D_S is computed each frame at time t in a compute shader. This map is then used as an exemplar input for the tiling and blending algorithm to create variety in the wave displacement and fully eliminate periodicity.

Following Equation 4, our model computes a final displacement map D from an initial time-dependent displacement map D_S as

$$D(\mathbf{x}, t) = \sum_{i=0}^{n-1} w_i(\mathbf{x}) \left(D_S\left(\mathbf{x} + \mathbf{h}\left(k_i(\mathbf{x})\right), t\right) - \mu_D(t)\right) + \mu_D(t), \tag{6}$$

where μ_D is the average displacement at time t. The ocean mesh is then displaced using Equation 1 as is.

The displacement D can be filtered by MIP-mapping the displacement map D_S at each step of time and executing the tiling and blending algorithm with the desired resolution.

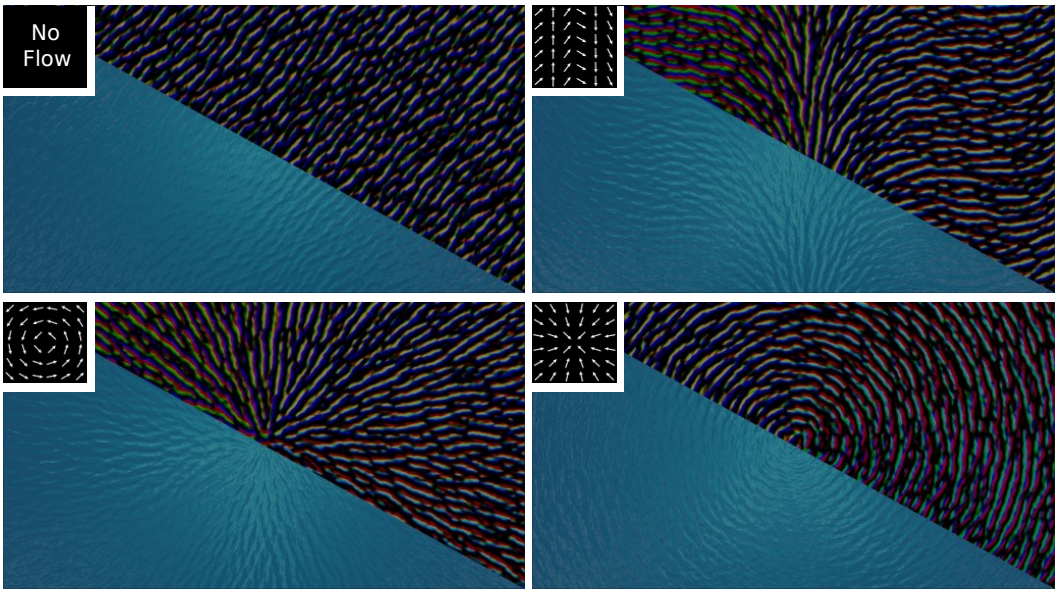

Fig. 6. Our synthesis model enables to author the orientation of the waves using a flow map. For each view: bottom left is a top down rendering of the ocean; top right is the corresponding visualization of the displacements; top left is a visualization of the flow map used.

Mean of the displacement map. When MIP-mapping D_S, the mean μ_D corresponds to the sole texel of the lowest resolution of D_S for each time step t. μ_D can often be deduced analytically: for instance, if D is computed from an Inverse Fourier Transform with amplitude A, μ_D is $A(\mathbf{0})$ and is constant with respect to time. In our experiments with several base spectra, we found the mean to be $\mu_D = \mathbf{0}$ at any step of time, which should be the case for most zero-mean trochoidal or sinusoidal displacement models.

4.3 Normal map

Normals can be computed by replacing D and $\mu_D(t)$ with N and $\mu_N(t)$ in Equation 6, and normalizing the result. If normals N are pre-filtered with a MIP-map, $\mu_N(t)$ can be computed the same way as μ_D by sampling the lowest resolution of N. In our experiments with the Tessendorf model, $\mu_N(t)$ closely corresponds to the normal of a still ocean at any step of time, which is the up vector.

To render specular highlights, we show how LEAN mapping [Olano and Baker 2010] can be used along with tiling and blending in Section 6.2.

5 WAVE ORIENTATION CONTROL

In this section, we propose to use a flow map to control the local orientations of waves for artistic authoring purposes. The proposed strategy exploits the underlying tiling and blending model to create coherent-looking waves: tiling is used to sample a single point per tile and avoid potential distortions of the displacement map D or the normal map N, while blending enables to smoothly transition from one wave direction to another.

We define a flow map $F : X \rightarrow \mathbb{R}^2$ which describes the desired direction of the waves at any plane mesh index $\mathbf{x} \in X$ as a set of normalized vectors. By default, wave spectra are computed by taking into account a normalized wind direction $\mathbf{w} \in \mathbb{R}^2$ provided as a user-defined parameter.

This wind direction controls the global direction in which the waves are traveling. To control the orientation of the synthesized waves using F, we propose to rotate them around the up vector by the signed angle $\theta(\mathbf{x})$ between $F(\mathbf{x})$ and $\mathbf{w}$:

$$\theta(\mathbf{x}) = \text{atan2}\left(F(\mathbf{x})\right) - \text{atan2}\left(\mathbf{w}\right), \tag{7}$$

which we use to define a rotation matrix $R^{(2)}_{\theta(\mathbf{x})}$, which rotates a vector on $\mathbb{R}^2$ (such as a plane mesh index in X) around a third orthogonal component, and $R^{(3)}_{\theta(\mathbf{x})}$, which rotates a vector on $\mathbb{R}^3$ (such as a displacement or a normal) around the up vector. A map giving the angles directly may also be pre-computed instead depending on the needs of the application (for instance if the wind direction is static or irrelevant).

There are two rotations that need to be executed when using a flow map:

- A rotation of the index $\mathbf{x}$ used to sample vectors, which enables the waves to stay coherent with time even when the wave direction is rotated;
- A rotation of any sampled vector, which enables the sampled displacements and normals to face the correct direction when sampled.

Using the tiling and blending model of Equation 6, we execute the following algorithm, for computing the displacement D for a given index $\mathbf{x}$:

(1) Compute the tiling functions k_i and the weighting functions w_i for position $\mathbf{x}$.
(2) Sample F at the center of each synthesis tile $k_i(\mathbf{x})$, which we call $c\left(k_i\left(\mathbf{x}\right)\right)$. We call this sample f.
(3) Rotate the offset index $\mathbf{x}+\mathbf{h}\left(k_i\left(\mathbf{x}\right)\right)$ around the origin by angle $\theta\left(f\right)$ using its rotation matrix $R^{(2)}_{\theta(f)}$, and sample the displacement map D_S at that position.
(4) Rotate the sampled displacement by the same angle $\theta\left(f\right)$ using $R^{(3)}_{\theta(f)}$ to correct the direction of the vector.

To summarize, following both Equation 6 and Equation 8, a tiling and blending of the displacement map is computed from a flow map F as

$$D(\mathbf{x},t) = \sum_{i=0}^{n-1} w_i\left(\mathbf{x}\right) R^{(3)}_{\theta(f)} \cdot \left(D_S\left(R^{(2)}_{\theta(f)} \cdot \left(\mathbf{x}+\mathbf{h}\left(k_i\left(\mathbf{x}\right)\right)\right), t\right) - \mu_D(t)\right) + \mu_D(t), \tag{8}$$

where all vectors in this equation are assumed to be row vectors. We show our results with different flow maps in Figure 6.

When trivially filtering the normal map N, one can use Equation 8 with N, N_S and μ_N instead of D, D_S and μ_D. Making this technique compatible with LEAN mapping, on the other hand, is not trivial. We discuss how to use a flow map with it in Section 6.3.

6 LEAN MAPPING

In this section, we show how to make LEAN mapping filtering [Olano and Baker 2010] compatible with our approach. We start by recalling how the original LEAN mapping algorithm works with a periodic normal map; then, by recalling how to make it compatible with tiling and blending. Finally, we propose a novel way to make it compatible with the flow map we use in Section 5.

6.1 LEAN mapping of a periodic normal map

To accurately filter the normal map for rendering coherent specular highlights, we rely on the LEAN mapping technique proposed by Olano et Baker [2010]. In this Beckmann-based shading model, each wave displacement at a given level is expressed as an off-centered Beckmann distribution in

the base ocean plane. To filter the specular highlights of the ocean at a given level, each normal distribution over a pixel footprint must be combined to compute the covariance matrix Σ of the global Beckmann distribution. In this shading model, the covariance matrix controls the specular lobe shape and size. As second-order moments combine linearly, a periodic map of the second-order moments of the displacements can be straightforwardly MIP-mapped to reconstruct the underlying shading distribution.

From a periodic normal map $N(\mathbf{x}) = (n.x, n.y, n.z)$, we compute two periodic maps

$$B(\mathbf{x}) = (\tilde{n.x}, \tilde{n.y}) \tag{9}$$

$$M(\mathbf{x}) = \left(\tilde{n.x^2}, \tilde{n.y^2}, \tilde{n.x\, n.y}\right) \tag{10}$$

where $\tilde{n.i} = \frac{n.i}{n.z}$. During rendering, both B and M are used to construct a covariance matrix Σ that depends on the position on the plane mesh. For any sampled value of B and M, that we call respectively b and m, regardless of the level of detail or the time,

$$\Sigma(\mathbf{x}) = \begin{pmatrix} M(\mathbf{x}).x - B(\mathbf{x}).x^2 + \frac{1}{s} & M(\mathbf{x}).z - B(\mathbf{x}).x\, B(\mathbf{x}).y \\ M(\mathbf{x}).z - B(\mathbf{x}).x\, B(\mathbf{x}).y & M(\mathbf{x}).y - B(\mathbf{x}).y^2 + \frac{1}{s} \end{pmatrix}, \tag{11}$$

where s represents the specular power of the ocean surface. We then reconstruct a specular lobe through the Beckmann distribution function of covariance matrix Σ [Olano and Baker 2010] for any index $\mathbf{x}$. Note that the model can be extended with LEADR mapping [Dupuy et al. 2013], which modulates the specular lobe by computing the masking-shadowing factor of displaced waves without requiring further pre-computations.

6.2 LEAN mapping of tiling and blending

Grenier et al. [2022] showed that using tiling and blending does not enable a straightforward LEAN mapping by only synthesizing both B and M using Equation 6, as synthesizing the covariance matrix elements requires to square the blending weights. LEAN mapping with tiling and blending can therefore be expressed from maps B and M by reconstructing a local covariance matrix for the i-th tile, and blending each of its elements with a squared weight.

Let us define a function $\sigma(b, m)$, which constructs the following matrix:

$$\sigma(b, m) = \begin{pmatrix} m.x - b.x^2 & m.z - b.x\, b.y \\ m.z - b.x\, b.y & m.y - b.y^2 \end{pmatrix}. \tag{12}$$

Following Equation 6 and using Equation 12, for any fixed level of detail and any fixed time, we express the final covariance matrix Σ at row r and column c as

$$\Sigma_{r,c}(\mathbf{x}) = \sum_{i=0}^{n-1} w_i^2\,(\mathbf{x})\left(\sigma(B_i(\mathbf{x}), M_i(\mathbf{x}))_{r,c} - \sigma(\mu_B, \mu_M)_{r,c}\right) + \sigma(\mu_B, \mu_M)_{r,c}\,, \tag{13}$$

where both B_i and M_i are constructed similarly to Equation 5 as

$$B_i(\mathbf{x}) = B\left(\mathbf{x} + \mathbf{h}\left(k_i\left(\mathbf{x}\right)\right)\right), \tag{14}$$

$$M_i(\mathbf{x}) = M\left(\mathbf{x} + \mathbf{h}\left(k_i\left(\mathbf{x}\right)\right)\right). \tag{15}$$

Note that similarly to Equation 6, μ_B and μ_M, which represent the means of respectively B and M, can be obtained by sampling B and M at the lowest resolution. Additionally, the term $\frac{1}{s}$ is added post-synthesis to $\Sigma_{0,0}(\mathbf{x})$ and $\Sigma_{1,1}(\mathbf{x})$.

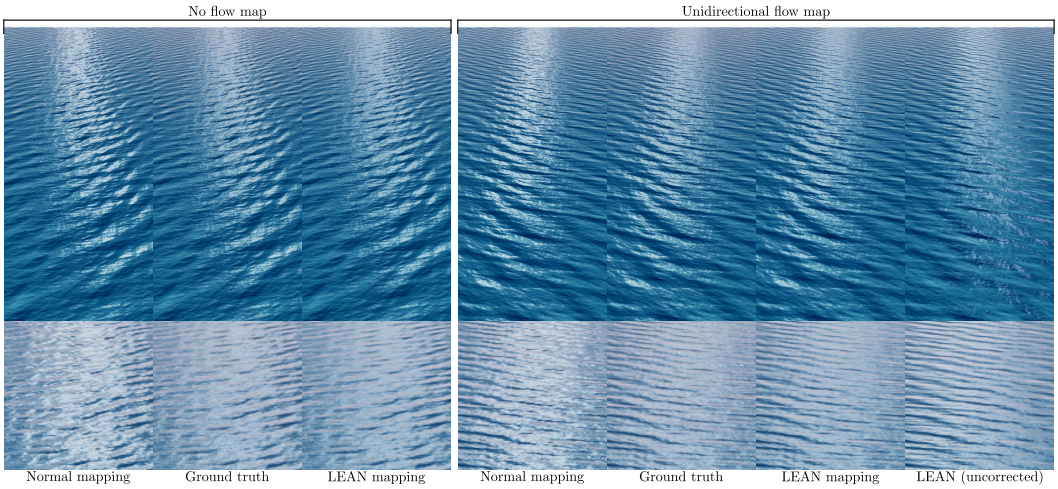

Fig. 7. Comparison between normal mapping, a ground truth, and LEAN mapping for rendering a specular lobe on our ocean. Left block: without a flow map. Right block: with a unidirectional flow map. Bottom of each view: zoom on the specular lobe at the horizon. Our correction allows to coherently LEAN map a surface synthesized with a tiling and blending whose contents are rotated. In this figure, the default wind direction goes towards the bottom right corner, while our flow map directs the waves towards the bottom left corner. Both the camera and time are fixed for each view.

6.3 LEAN mapping with a flow map

LEAN mapping requires another adjustment to be used alongside flow maps in the context of wave orientation control, which we discussed in Section 5. An issue that arises when using a flow map is that all normals need to be rotated at the highest resolution. However, their local first-order moments and second-order moments, stored in B and M respectively, are pre-filtered without rotation.

Values sampled from map B can be trivially rotated at any level of detail around the up vector, because any sampled value of B is a linear combination of vectors. Therefore, following the notations of Grenier et al. [2022], the rotation-corrected MIP-map B is expressed for any footprint $\mathbb{P}$ as

$$B_\phi(\mathbb{P}) = R_\phi^{(2)} B(\mathbb{P})\,. \tag{16}$$

Values sampled from M are pre-filtered products of normals, which do not enable a straightforward rotation: we want to compute the product of the rotated values, which is not the same as rotating the product of the values. Fortunately, we show that we can express each component of a rotation-corrected map M_ϕ according to a rotation angle ϕ as a linear combination of components of M, which allows us to reconstruct the proper rotation at render time.

For the rotation-corrected variance of the normals on either axis, stored in $M.x$ and $M.y$, we need to compute

$$M_\phi.xy(\mathbb{P}) = \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} R_\phi^{(2)} B(\mathbf{x}) \odot R_\phi^{(2)} B(\mathbf{x}) \tag{17}$$

and

$$M_\phi.z(\mathbb{P}) = \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} \left(R_\phi^{(2)} B(\mathbf{x})\right).x \left(R_\phi^{(2)} B(\mathbf{x})\right).y\,, \tag{18}$$

where $\#\mathbb{P}$ is the amount of texels that fall within the footprint $\mathbb{P}$ and $\odot$ is the pointwise product. Equations 17 and 18 are too expensive to be computed in real-time, and cannot be pre-filtered for all possible angles ϕ. We show in appendix A that we can actually express each component of M_ϕ relative to M for any footprint $\mathbb{P}$:

$$M_\phi.x(\mathbb{P}) = \cos^2(\phi)M.x(\mathbb{P}) + \sin(2\phi)M.z(\mathbb{P}) + \sin^2(\phi)M.y(\mathbb{P}) \tag{19}$$

$$M_\phi.y(\mathbb{P}) = \sin^2(\phi)M.x(\mathbb{P}) - \sin(2\phi)M.z(\mathbb{P}) + \cos^2(\phi)M.y(\mathbb{P}) \tag{20}$$

$$M_\phi.z(\mathbb{P}) = -\frac{1}{2}\sin(2\phi)M.x(\mathbb{P}) + \cos(2\phi)M.z(\mathbb{P}) + \frac{1}{2}\sin(2\phi)M.y(\mathbb{P})\,, \tag{21}$$

allowing any MIP-map texel of the rotation-corrected map M_ϕ to be computed on the fly with ease only by MIP-mapping M.

For using our rotation-corrected LEAN mapping with the tiling and blending of Equation 13 alongside the rotation of the index in Equation 8, both B_i and M_i of Equation 14 must be replaced with their rotation-corrected equivalent. We show the effect of our correction compared to an uncorrected LEAN and a corrected normal mapping in Figure 7.

When attempting to filter the surface with anisotropic filtering in a fragment shader, note that one needs to rotate the fragment derivatives in x and y with matrix $R_\phi^{(2)}$ as well.

7 RESULTS

In this section, we show results in terms of both visual appearance and performance. All the oceans synthesized in this section have been generated using a JONSWAP spectrum [Horvath 2015], whose parameters are a wave amplitude $a = 0.05$, $\gamma = 3.3$ and a fetch $F = 200m$. We also used a swell $\xi = 0.9$ to create elongated waves. Our spectrum is computed using the Hasselmann directional spreading model, and a dispersion relationship corresponding to the deep water approximation. In all our examples, the initial ocean tile used has a dimension of $250m \times 250m$. All visual results in this work were created using an initial ocean tile size of 512×512. For tiling and blending, we used a synthesis tile size that corresponds to two initial ocean tile periods from one vertex of the hexagon to its opposite vertex. We motivate this high tiling size in section 7.1.

7.1 Visual quality

We assess the quality of our ocean synthesis by studying the autocorrelation function (ACf) of its displacement, which can be expressed as the IFFT of the power spectrum [Lagae et al. 2010], yet can be estimated in the spatial domain. We then compare the different displacement generation methods of Section 3.1 and ours (Section 4.2) to the expected ACf of a regular ocean displacement.

The ACf is a measure of the spatial correlation of content between indices separated by a translation τ. It is computed from the autocovariance function of the displacement, which is exactly expressed as ([Lutz et al. 2023]):

$$\widetilde{r}_D(\tau) = \frac{1}{|X|}\int_X (D(\mathbf{x}) - \widetilde{\mu_D}) \odot (D(\mathbf{x}+\tau) - \widetilde{\mu_D})\; d\mathbf{x}, \tag{22}$$

where $\widetilde{\mu_D}$ is the estimation of the mean of the displacement. Following Equation 22, the estimation of the ACf is $\frac{\widetilde{r}_D(\tau)}{\widetilde{r}_D(0)}$. Because D is infinite, we estimate D over a large number of random values of $\mathbf{x}$.

Yang et al. [2020] show that the ACf of an ocean approaches zero as time increases for both the JONSWAP spectrum [Horvath 2015] and the Pierson-Moskowitz spectrum [Pierson Jr. and Moskowitz 1964]. Meanwhile, the Tessendorf model [Tessendorf 2001] yields a spatially-periodic

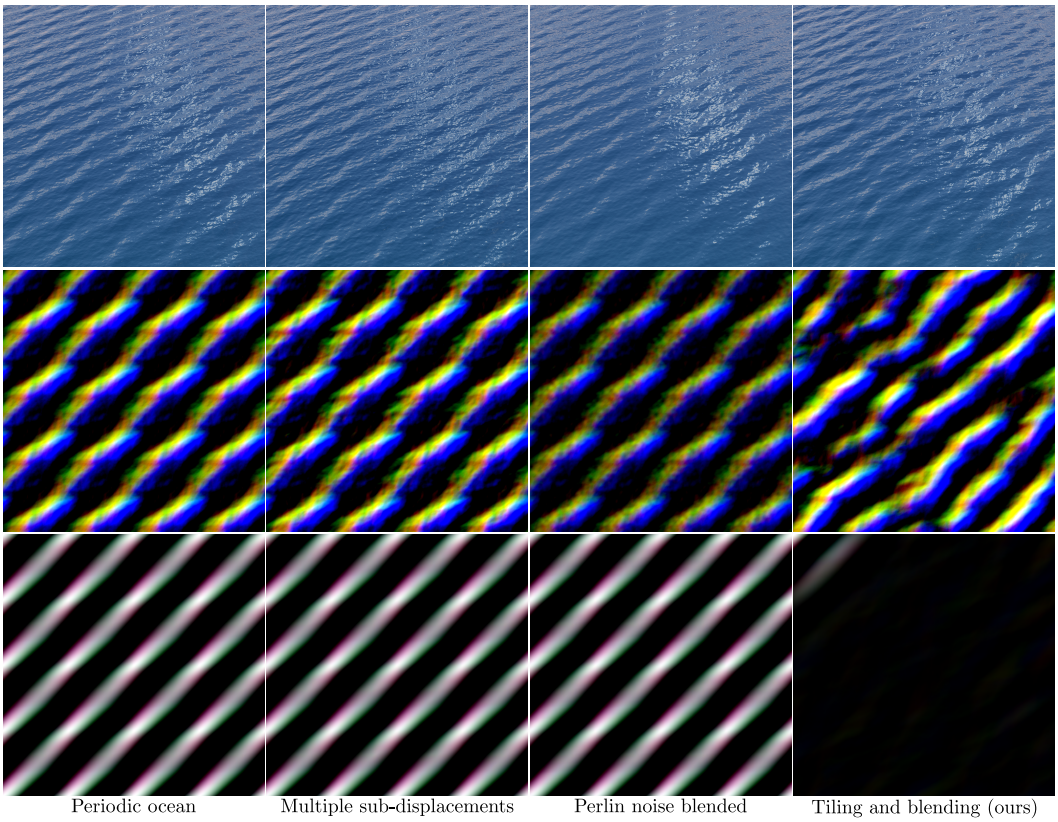

Fig. 8. From top to bottom: ocean rendering (top), displacement map D over 4 times the period of the periodic ocean (middle) and their positive autocorrelation (bottom), with four ocean rendering algorithms: periodic ocean [Tessendorf 2001], 3 overlapping sub-displacements [Deliot and Heitz 2018], periodic ocean blended with low frequency Perlin noise [NVIDIA 2011] and ours using tiling and blending [Heitz and Neyret 2018]. The autocorrelation of ours is the only one that decreases to 0 as its parameter t increases. Sub-displacements scaled at (respectively) 1x, 0.84x and 0.52x the period for low, middle and high frequencies. Perlin noise weight is 33% of the total displacement. The size of one hexagonal tile is two initial ocean tile periods.

ocean displacement, meaning that its ACf is periodic as well. Using multiple sub-displacements as in Equation 3 creates a periodic ocean, so the ACf is periodic, albeit with a potentially extremely large period. In our experiments, the ACf is still periodically high due to the different sub-displacements being themselves periodic. This is especially noticeable for low frequency waves, which had the highest impact on the ACf in our experiments. Blending the periodic ocean with a spatially-stationary noise as in Equation 2 creates a periodicity in the ACf after a full period, because it creates a spatially-cyclostationary ocean.

As opposed to other methods, tiling and blending yields an ocean displacement whose ACf is decreasing due to the content of synthesis tiles being picked randomly and uniformly, as shown by Lutz et al. [2023] who attempt to reestablish it. In other words, this potentially undesired effect is instead an advantage of our application. We show this effect for our method and the others in Figure 8.

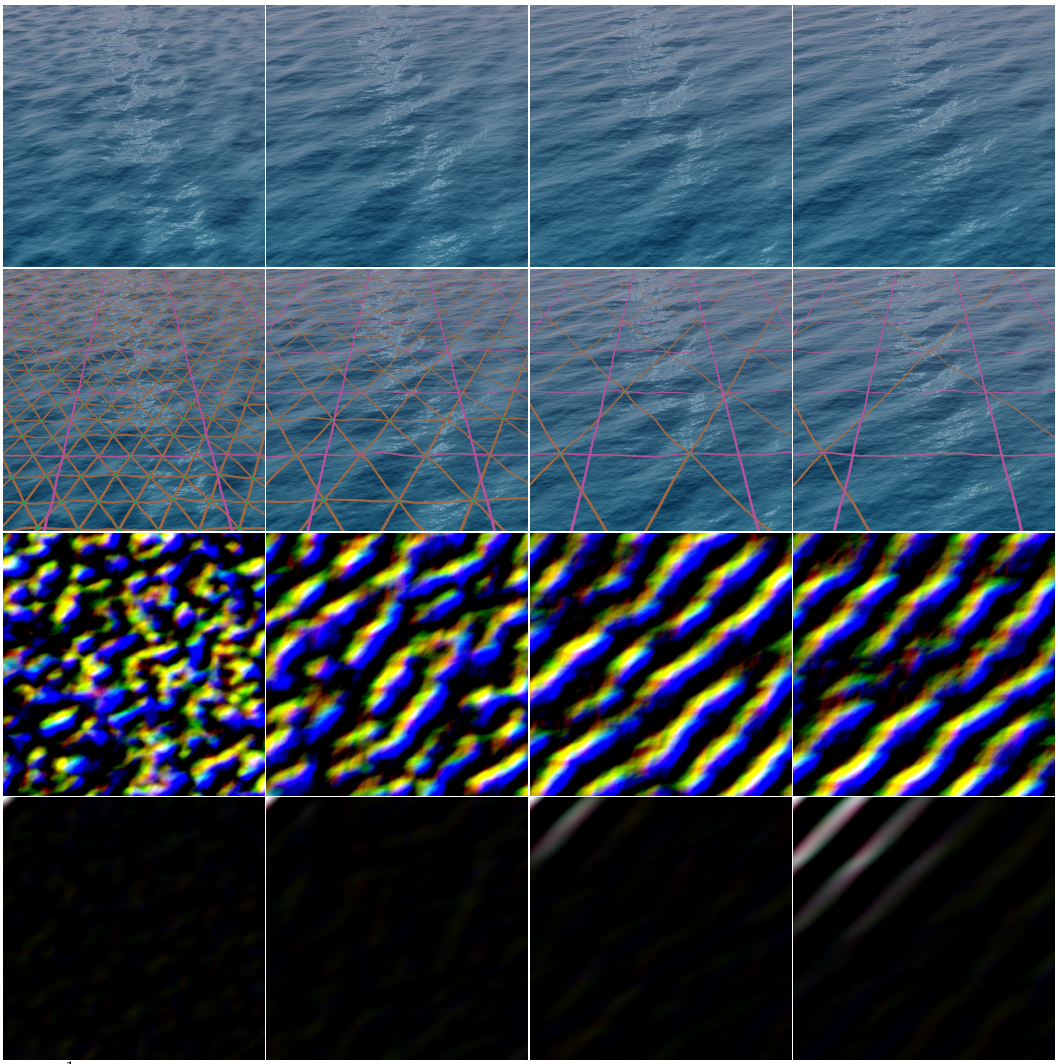

Fig. 9. From top to bottom: Ocean rendering, visualization of the initial ocean tile (pink) and the synthesis tiling (orange), displacement map D over four times the period of the initial ocean tile, and positive autocorrelation of the displacement. Our method is showcased with synthesis tiling sizes varying from half to four times the period of the initial ocean tile (from one edge of a hexagon to its opposite edge). The size of the synthesis tilings determines the distance and rate at which the autocorrelation decreases. A tiling size too small introduces low frequencies due to rapid changes in tile contents, while a tiling size too high reestablishes local periodicities.

Controlling the preservation of the ACf. As shown by Lutz et al. [2023], the autocovariance function is preserved within the size of a single synthesis tile, where it decreases towards 0 at the edge of the tile. This means that the tile size is directly proportional to the distance at which the autocorrelation is well-preserved, and that modifying the tile size is a mean to author this distance.

A high synthesis tile size reestablishes local periodicities because each tile includes more than the entire periodic ocean, while a low tile size creates high frequency waves due to rapid shifts

	Periodic ocean	Multiple sub-displacements	Perlin noise blended	Ours
128×128				
Tile generation	0.78 ms	2.38 ms	0.78 ms	0.78 ms
Rendering	1.52 ms	1.73 ms	1.64 ms	1.65 ms
Total GPU time	2.30 ms	4.11 ms	2.42 ms	2.43 ms
256×256				
Tile generation	0.98 ms	2.98 ms	0.98 ms	0.98 ms
Rendering	1.49 ms	1.82 ms	1.59 ms	1.78 ms
Total GPU time	2.47 ms	4.80 ms	2.57 ms	2.76 ms
512×512				
Tile generation	2.16 ms	6.47 ms	2.16 ms	2.16 ms
Rendering	1.45 ms	2.03 ms	1.52 ms	1.79 ms
Total GPU time	3.61 ms	8.50 ms	3.68 ms	3.95 ms

Table 1. Computation times using the various methods showcased in Figure 9, for the initial ocean tile computation and the rendering cost, for various ocean tile sizes, without LEAN mapping and without using a flow map. Our method slightly increases the cost of rendering for the benefit of creating true aperiodicity, and avoids the computation of several sub-displacements.

between one part of the ocean and the next. In practice, we choose to keep the size of the tiles in our synthesis rather high: this enables us to preserve low distance correlations, which matters the most to keep the global shape of ocean waves, and keep the stationarity of the ocean at higher distances. We show the effect of this parameter on the ACf in Figure 9.

7.2 Computation time

Table 1 showcases the time measurements captured using our unoptimized prototype on a NVIDIA GeForce RTX 2070 SUPER. In this table, we compare the cost of generating and rendering an ocean using a single periodic displacement map D with 3 periodic sub-displacement maps D_i and our synthesized displacement map, using the IFFT of an oceanographic spectrum to model displacements, without LEAN mapping and without using a flow map. The results shown in this paper have been generated using a HLSL implementation of our method, using HLSL computer shaders to perform efficient computations on the GPU. The rendering time corresponding to our method is close to other methods, showing that tiling and blending results in a small computation time overhead in our application. We found the multiple texture accesses of tiling and blending to be the most costly operation in our application compared to other methods. Using the two texture access tiling and blending of Lutz et al. [2021] could lower this cost, but at the expense of generating visible singularities.

Computing 3 periodic sub-displacement maps multiplies by 3 the number of IFFTs to compute per frame, effectively increasing the cost of an ocean update by a factor close to 3, and further increases the cost in the fragment shader due to sampling multiple textures. We note that the ocean tile generation for sub-displacements could theoretically be optimized by cutting computations down when the different bands of frequency do not overlap, although it still requires writing and sampling multiple textures each frame.

Flow map. Using a flow map requires to sample an unfiltered texture on the ocean surface three times, which slightly increases rendering costs. In our experiments, we observed that using a flow

map resulted in a rendering overhead of 0.1 ms on average in our examples. Although slightly more expensive, our method remains cheaper than computing and blending 4 oceans [Olano and Baker 2010], enabling unprecedented performance using a flow map.

LEAN mapping. With tiling and blending, the LEAN mapping of section 6.2 requires 6 more texture accesses as it requires synthesizing both maps B and M. It is also more expensive than LEAN mapping a periodic normal map for the same reason. To put this into perspective, we note that it is not possible to LEAN map the periodic ocean with Perlin noise, since B and M cannot be precomputed. Moreover, LEAN mapping multiple sub-displacements requires to compute the LEAN mapping for each sub-displacement, increasing both ocean tile generation cost and rendering cost. The performance overhead coming with LEAN mapping is about 1.4 ms with an ocean tile of size 512×512 using the same parameters as Table 1. In practice, as shown by Olano and Baker [2010], this cost can be mitigated by placing the normal map inside the LEAN map, lowering the number of texture accesses required for filtered shading by three.

7.3 Memory consumption

Our technique does not have increased memory consumption besides a few uniform variables that control the synthesis, such as tiling scale or tiling offset. Compared to the generation of several sub-displacement maps D_i, our method effectively decreases the virtual memory cost of displacements and normals by generating only one displacement map and one normal map per frame.

8 DISCUSSION

In this section, we discuss implementation details and present limitations of our work.

Repetition artifacts. Tiling and blending is known to produce repetition artifacts as a direct consequence of using tiles of existing content. For our ocean simulation, this means that two synthesis tiles that fetch the same content will always have similar looking waves at any point of time. In practice, because tiles are blended together, this effect is not particularly noticeable. Furthermore, the variety our synthesis is capable of achieving is directly tied to the amount of samples in the initial ocean tile computed by the IFFT. This could be improved in the future by enhancing the computation speed and size of the initial ocean tile, or by storing and sampling time-adjacent ocean tiles.

Artistic control. An advantage of the system with sub-displacement maps proposed by Dupuy and Bruneton [2012] is that an artist can author the ocean by controlling both the period scale and the value scale of each sub-displacement map D_i. While this does not result in a fully aperiodic ocean, it gives a mean to control the look of the ocean through the control of its sub-displacements. Our synthesis removes this ability to control overlapping displacements of the final ocean because the initial displacement map D_S is computed from the entire input spectrum S.

However, our method still offers control over the generated appearance: the initial spectrum can be controlled to alter the final result, and the size of the synthesis tiles, the offset of the tilings and the seeding algorithm can be changed. Ultimately, the sub-displacement system of Dupuy and Bruneton [2012] is itself compatible with our synthesis by taking any or all sub-displacement maps D_i as input of Equation 6. This can be used to remove the apparent periodicity of any sub-displacement map D_i while keeping the artistic control enabled by tweaking sub-displacement maps individually, at the expense of its increased computational and memory cost.

Histogram preservation. Tiling and blending is known to increase the fit of the first-order distribution of the output with a multivariate Gaussian distribution compared to the exemplar. Heitz

and Neyret [2018] proposed a real-time histogram transfer to better preserve the histogram of the exemplar. It requires the pre-computations of a "Gaussianized" version of the exemplar, the histogram of the exemplar and their corresponding inverse histogram transfer into a lookup table for each resolution [Deliot and Heitz 2018]. Using this technique in our application would require to execute these pre-computations each frame, which would significantly increase computation times. Furthermore, the causal relationship between the synthesized displacement map and the synthesized normal map would have to be preserved with care throughout the Gaussianization and the inverse transfer, which is potentially not trivial since these maps are synthesized in different parts of the graphics pipeline. For these reasons, we choose to only trivially preserve the variance. In the future, we think it would be interesting to approximate a preservation of the histogram, for instance, by pre-computing a static histogram that would well match the time-varying histogram of our ocean on average.

9 CONCLUSION AND FUTURE WORKS

In this work, we present a new way to efficiently and aperiodically tile an ocean with minimal alteration of an input ocean spectrum using a state-of-the-art texture synthesis algorithm. Our approach is faster to compute than overlapping multiple sub-displacements computed from sub-IFFTs, better preserves ocean features than blending the displacement with Perlin noise, and yields an autocorrelation function closer to that of a real ocean in both cases. Our method also offers artistic control through the use of a user-defined flow map to author the orientations of the waves of the ocean. Finally, we show how to filter the specular highlights of our ocean with LEAN mapping in all scenarios.

In future works, we would like to enhance the performances of the generation of the initial displacement map. We could start by exploring the various noises controlled in the spectral domain by a spectrum, which efficiently produce maps with the characteristics of that spectrum [Galerne et al. 2012; Gilet et al. 2014]. We would also like to explore the optimization of the parameters of tiling and blending for getting an ACf that fits that of a realistic ocean displacement [Yang et al. 2020]. We think that tiling and blending could also be further explored in the context of by-example synthesis of animated textures such as advected textures [Neyret 2003] using both the synthesis and its flow map. Finally, we would like to increase the authoring capabilities of our ocean, for instance by proposing a spatial variation of the amplitude of the waves.

ACKNOWLEDGMENTS

This work was supported by the Mitacs Accelerate Program, the University of Sherbrooke, and Ubisoft Inc. We thank Arthur Delon and Josué Raad for their expertise regarding LEAN mapping. We also thank Hisanari Otsu, Damien Rioux-Lavoie, Antoine Houdard and Georges Nader for their feedbacks on the early version of our paper.

REFERENCES

Robert Bridson. 2015. *Fluid simulation for computer graphics.* CRC press. https://doi.org/10.1201/9781315266008

Brent Burley. 2019. On Histogram-Preserving Blending for Randomized Texture Tiling. *Journal of Computer Graphics Techniques (JCGT)* 8, 4 (8 November 2019), 31–53. http://jcgt.org/published/0008/04/02/

Yung-Feng Chiu and Chun-Fa Chang. 2006. GPU-based ocean rendering. In *2006 IEEE international conference on multimedia and expo*. IEEE, 2125–2128. https://doi.org/10.1109/ICME.2006.262655

Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. 2003. Wang Tiles for Image and Texture Generation. *ACM Transactions on Graphics* 22, 3 (2003), 287–294. https://doi.org/10.1145/882262.882265

Emmanuelle Darles, Benoît Crespin, Djamchid Ghazanfarpour, and Jean-Christophe Gonzato. 2011. A survey of ocean simulation and rendering techniques in computer graphics. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 43–60. https://doi.org/10.1111/j.1467-8659.2010.01828.x

Thomas Deliot and Eric Heitz. 2018. Procedural Stochastic Textures by Tiling and Blending. *GPU Zen 2: Advanced Rendering Techniques* (2018).

Jonathan Dupuy and Eric Bruneton. 2012. Real-time animation and rendering of ocean whitecaps. In *SIGGRAPH Asia 2012 Technical Briefs* (Singapore, Singapore) *(SA '12)*. Association for Computing Machinery, New York, NY, USA, Article 15, 3 pages. https://doi.org/10.1145/2407746.2407761

Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. 2013. Linear efficient antialiased displacement and reflectance mapping. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–11.

Jocelyn Fréchot. 2006. Realistic simulation of ocean surface using wave spectra. In *Proceedings of the first international conference on computer graphics theory and applications (GRAPP 2006)*. 76–83. https://hal.science/hal-00307938

Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. 2011. Random Phase Textures: Theory and Synthesis. *IEEE Transactions on Image Processing* 20, 1 (2011), 257 – 267. https://doi.org/10.1109/TIP.2010.2052822

Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. 2012. Gabor Noise by Example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)* 31, 4 (July 2012), 73:1–73:9. https://doi.org/10.1145/2185520.2335424

Bruno Galerne, Arthur Leclaire, and Lionel Moisan. 2017. Texton Noise. *Computer Graphics Forum* 36, 8 (2017), 205–218. https://doi.org/10.1111/cgf.13073

Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. 2014. Local Random-phase Noise for Procedural Texturing. *ACM Trans. Graph.* 33, 6, Article 195 (Nov. 2014), 11 pages. https://doi.org/10.1145/2661229.2661249

Charline Grenier, Basile Sauvage, Jean-Michel Dischler, and Sylvain Thery. 2022. Color-mapped noise vector fields for generating procedural micro-patterns. In *Computer Graphics Forum*, Vol. 40. https://doi.org/10.1111/cgf.14693

Eric Heitz and Fabrice Neyret. 2018. High-Performance By-Example Noise using a Histogram-Preserving Blending Operator. *Eurographics Symposium on High-Performance Graphics 2018* (2018). https://doi.org/10.1145/3233304

Christopher J. Horvath. 2015. Empirical directional wave spectra for computer graphics. In *Proceedings of the 2015 Symposium on Digital Production* (Los Angeles, California) *(DigiPro '15)*. Association for Computing Machinery, New York, NY, USA, 29–39. https://doi.org/10.1145/2791261.2791267

Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, D.S. Ebert, J.P. Lewis, Ken Perlin, and Matthias Zwicker. 2010. State of the Art in Procedural Noise Functions. In *EG 2010 - State of the Art Reports*, Helwig Hauser and Erik Reinhard (Eds.). Eurographics, Eurographics Association. http://www-sop.inria.fr/reves/Basilic/2010/LLCDDELPZ10

Namkyung Lee, Nakhoon Baek, and Kwan Woo Ryu. 2007. Real-time simulation of surface gravity ocean waves based on the tma spectrum. In *Computational Science–ICCS 2007: 7th International Conference, Beijing, China, May 27-30, 2007, Proceedings, Part II 7*. Springer, 122–129. https://doi.org/10.1007/978-3-540-7

Nicolas Lutz, Basile Sauvage, and Jean-Michel Dischler. 2021. Cyclostationary Gaussian noise: theory and synthesis. In *Eurographics 2021*. Vienna, Austria. https://hal.archives-ouvertes.fr/hal-03181139

Nicolas Lutz, Basile Sauvage, and Jean-Michel Dischler. 2023. Preserving the Autocovariance of Texture Tilings Using Importance Sampling. *Computer Graphics Forum* (2023). https://doi.org/10.1111/cgf.14766

Jason L Mitchell. 2005. Real-time synthesis and rendering of ocean water. *ATI Research Technical Report* 4, 1 (2005), 121–126. https://citeseerx.ist.psu.edu/document?doi=00478af7044a7f1350d5ec75ffc7c15b40057051

Fabrice Neyret. 2003. Advected textures. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, Eurographics Association, 147–153. https://dl.acm.org/doi/10.5555/846276.846297

NVIDIA. 2011. Ocean Surface Simulation. In *NVIDIA Graphics SDK 11 Direct3D*. https://developer.download.nvidia.com/assets/gamedev/files/sdk/11/OceanCS_Slides.pdf

Marc Olano and Dan Baker. 2010. LEAN mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 181–188. https://doi.org/10.1145/1730804.1730834

Willard J. Pierson Jr. and Lionel Moskowitz. 1964. A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. *Journal of Geophysical Research (1896-1977)* 69, 24 (1964), 5181–5190. https://doi.org/10.1029/JZ069i024p05181

Björn Rydahl. 2009. *A VFX ocean toolkit with real time preview*. Master's thesis. http://www.diva-portal.org/smash/get/diva2:272268/FULLTEXT02

Jerry Tessendorf. 2001. Simulating Ocean Water. SIGGRAPH 2001 Course notes.

Mária Vámošová. 2014. *Wang Tiles for Frequency-Based Synthesis of Ocean Surfaces*. Master's thesis. Univerzita Karlova, Matematicko-fyzikální fakulta.

Kenneth Vanhoey, Basile Sauvage, Frédéric Larue, and Jean-Michel Dischler. 2013. On-the-Fly Multi-Scale Infinite Texturing from Example. *Transactions on Graphics* 32, 6 (2013), 208:1–208:10. https://doi.org/10.1145/2508363.2508383 (Proceedings of Siggraph Asia'13).

Wen Yang, Yuke Liang, Jianxing Leng, and Ming Li. 2020. The Autocorrelation Function Obtained from the Pierson-Moskowitz Spectrum. In *Global Oceans 2020: Singapore – U.S. Gulf Coast*. 1–4. https://doi.org/10.1109/IEEECONF38699.2020.9389043

A DETAILS FOR THE ROTATION-CORRECTED LEAN MAPPING

In Section 4.3, we gave the final form of the rotation-corrected LEAN mapping for maps B and M, the latter being non-trivial to compute.

Recall that we need to compute the rotation-corrected variances

$$M_\phi.xy(\mathbb{P}) = \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} R_\phi^{(2)} B(\mathbf{x}) \odot R_\phi^{(2)} B(\mathbf{x}),$$

where

$$R_\phi^{(2)} = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix}.$$

When developing the expression and simplifying $\cos(\phi)\sin(\phi)$ to $\frac{1}{2}\sin(2\phi)$, they become

$$M_\phi.x(\mathbb{P}) = \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} \cos^2(\phi)\, B.x^2(\mathbf{x}) + \sin(2\phi)\, B.x(\mathbf{x})\, B.y(\mathbf{x}) + \sin^2(\phi)\, B.y^2(\mathbf{x})$$

and

$$M_\phi.y(\mathbb{P}) = \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} \sin^2(\phi)\, B.x^2(\mathbf{x}) - \sin(2\phi)\, B.x(\mathbf{x})\, B.y(\mathbf{x}) + \cos^2(\phi)\, B.y^2(\mathbf{x}).$$

Through commutations and distributions, these variances can be brought to

$$\begin{aligned} M_\phi.x(\mathbb{P}) &= \cos^2(\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.x^2(\mathbf{x}) \;+\; \sin(2\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.x(\mathbf{x})\, B.y(\mathbf{x}) \;+\; \sin^2(\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.y^2(\mathbf{x}) \\ &= \cos^2(\phi)\, M.x(\mathbb{P}) + \sin(2\phi)\, M.z(\mathbb{P}) + \sin^2(\phi)\, M.y(\mathbb{P}) \end{aligned}$$

and

$$\begin{aligned} M_\phi.y(\mathbb{P}) &= \sin^2(\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.x^2(\mathbf{x}) \;-\; \sin(2\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.x(\mathbf{x})\, B.y(\mathbf{x}) \;+\; \cos^2(\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.y^2(\mathbf{x}) \\ &= \sin^2(\phi)\, M.x(\mathbb{P}) \;-\; \sin(2\phi)\, M.z(\mathbb{P}) \;+\; \cos^2(\phi)\, M.y(\mathbb{P}). \end{aligned}$$

The rotation-corrected covariance is

$$M_\phi.z(\mathbb{P}) = \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} \left(R_\phi^{(2)} B(\mathbf{x})\right).x \left(R_\phi^{(2)} B(\mathbf{x})\right).y\ .$$

When developing this expression, simplifying the sum of $\cos^2$ and $\sin^2$ to $\cos(2\phi)$, and simplifying $\cos(\phi)\sin(\phi)$ to $\frac{1}{2}\sin(2\phi)$, it becomes

$$M_\phi.z(\mathbb{P}) = \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} \;-\frac{1}{2}\sin(2\phi)\, B.x^2(\mathbf{x}) \;+\; \cos(2\phi)\, B.x(\mathbf{x})\, B.y(\mathbf{x}) \;+\; \frac{1}{2}\sin(2\phi)\, B.y^2(\mathbf{x}),$$

Once again, through commutations and distributions, the covariance can be brought to

$$\begin{aligned} M_\phi.z(\mathbb{P}) &= -\frac{1}{2}\sin(2\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.x^2(\mathbf{x}) \;+\; \cos(2\phi)\, \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.x(\mathbf{x})\, B.y(\mathbf{x}) \\ &\quad + \frac{1}{2}\sin(2\phi)\; \frac{1}{\#\mathbb{P}} \sum_{\mathbf{x}\in\mathbb{P}} B.y(\mathbf{x})^2 \\ &= -\frac{1}{2}\sin(2\phi)\, M.x(\mathbb{P}) \;+\; \cos(2\phi)\, M.z(\mathbb{P}) \;+\; \frac{1}{2}\sin(2\phi)\, M.y(\mathbb{P}). \end{aligned}$$

B OCEAN TILE GENERATION

This section describes the computation of the initial, time-dependant ocean tile, used as exemplar in our method. We closely follow the computation models of Tessendorf [2001] and Horvath [2015]. Given a discrete coordinate $\mathbf{x}$ in the discrete index set of the ocean tile, we compute its corresponding complex ocean displacement vector via an IFFT:

$$h(\mathbf{x}, t) = \sum_{\mathbf{k}} \hat{h}(\mathbf{k}, t) e^{i\mathbf{k}^T\mathbf{x}}, \tag{23}$$

with $\mathbf{k} = (k_x, k_y)$ being wavevectors, with $k_x = 2\pi n/L_x$ and $k_y = 2\pi m/L_y$; n and m are discrete coordinates such that $-N/2 \leqslant n < N/2$ and $-M/2 \leqslant m < M/2$ with $N \times M$ the resolution of the grid used, and $L_x \times L_y$ are the dimensions of the ocean tile. The displacement map D of Equation 1 is the real part of Equation 23.

Given a dispersion relationship $\omega(k)$, the Fourier amplitudes of the wave field realization at time t are computed using:

$$\hat{h}(\mathbf{k}, t) = \hat{h}_0(\mathbf{k}) e^{i\omega(k)t} + \hat{h}_0^*(-\mathbf{k}) e^{-i\omega(k)t}, \tag{24}$$

with the Fourier amplitude of a wave height field $\hat{h}_0(\mathbf{k})$ defined as

$$\hat{h}_0(\mathbf{k}) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{2\mathcal{S}(\omega)\mathcal{D}(\omega, \theta)\frac{\partial\omega}{\partial k}\frac{1}{k}\Delta k_x \Delta k_y}\,, \tag{25}$$

where ξ_r and ξ_i are numbers computed using a Gaussian random number generator with mean 0 and standard deviation 1, $\theta = \operatorname{atan}(\frac{k_y}{k_x})$ corresponds to the angle of the wave relative to the wind direction, $\mathcal{S}(\omega)$ is a non-directional wave spectrum, and $\mathcal{D}(\omega, \theta)$ is a directional spreading function. The majority of our results have been generated using the JONSWAP wave spectrum:

$$\mathcal{S}(\omega) = \frac{\alpha g^2}{\omega^5} \exp(-\frac{5}{4}(\frac{\omega_p}{\omega})^4)\gamma^4\,, \tag{26}$$

with $\alpha = 0.076(\frac{U^2}{Fg})^{0.22}$, $\omega_p = 22\frac{g^2}{UF}$, $\gamma = 3.3$, U the average wind speed, F the fetch, $g = 9.81$ is the gravitational constant, and $\omega_p = 0.855g/U$ the peak frequency. We used a directional spreading function integrating a swell parameter introduced by Horvath [2015] to generate all the results in this paper. This function is defined as

$$\mathcal{D}(\omega, \theta) = Q(s)\,|\cos(\theta/2)|^{2s}\,, \tag{27}$$

with

$$Q(s) = \frac{2^{2s-1}}{\pi}\frac{\Gamma(s+1)^2}{\Gamma(2s+1)}, \tag{28}$$

and

$$s = 16\tanh(\frac{\omega_p}{\omega})\xi^2. \tag{29}$$

In Equation 28, Γ is the Euler gamma function, and ξ is the swell parameter. Note that for all the results generated in this paper, we used a dispersion relationship corresponding to the deep water approximation $\omega(k) = \sqrt{gk}$ and $\frac{\partial\omega(k)}{\partial k} = \frac{g}{2\sqrt{gk}}$.

Following Tessendorf [2001] and for shading purposes, we compute an ocean normal map N from local slope vectors $\epsilon(\mathbf{x}, t)$:

$$N(\mathbf{x}, t) = \frac{(-\epsilon_x(\mathbf{x}, t), -\epsilon_y(\mathbf{x}, t), 1)}{\left\|(-\epsilon_x(\mathbf{x}, t), -\epsilon_y(\mathbf{x}, t), 1)\right\|}. \tag{30}$$

In practice, the slope vector can be exactly computed using additional IFFTs:

$$\epsilon(\mathbf{x}, t) = \nabla h(\mathbf{x}, t) = \sum_{\mathbf{k}} i\mathbf{k}\hat{h}(\mathbf{k}, t)e^{i\mathbf{k}^T\mathbf{x}}. \tag{31}$$