Synthesizing Waves from Animated Height Fields

MICHAEL B. NIELSEN
Weta Digital and Aarhus University
ANDREAS SÖDERSTRÖM
Weta Digital
and
ROBERT BRIDSON
Weta Digital and University of British Columbia

Computer animated ocean waves for feature films are typically carefully choreographed to match the vision of the director and to support the telling of the story. The rough shape of these waves is established in the previsualization (previs) stage, where artists use a variety of modeling tools with fast feedback to obtain the desired look. This poses a challenge to the effects artists who must subsequently match the locked-down look of the previs waves with high-quality simulated or synthesized waves, adding the detail necessary for the final shot. We propose a set of automated techniques for synthesizing Fourier-based ocean waves that match a previs input, allowing artists to quickly enhance the input wave animation with additional higher-frequency detail that moves consistently with the coarse waves, tweak the wave shapes to flatten troughs and sharpen peaks if desired (as is characteristic of deep water waves), and compute a physically reasonable velocity field of the water analytically. These properties are demonstrated with several examples, including a previs scene from a visual effects production environment.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Physically based modeling*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Fluid modeling, fluid control, animation, fluid simulation, physically based animation

Authors' addresses: M. B. Nielsen (corresponding author), Aarhus University, Aabogade 34, 8200 Aarhus N, Denmark; email: nielsenmb@gmail.com; A. Söderström, Weta Digital, 9-11 Manuka Street, Miramar, Wellington, New Zealand 6022; R. Bridson, University of British Columbia, 201–2366 Main Mall, Vancouver, BC, Canada V6T 1Z4.

DOI 10.1145/2421636.2421638
http://doi.acm.org/10.1145/2421636.2421638

ACM Reference Format:
Nielsen, M. B., Söderström, A., and Bridson, R. 2013. Synthesizing waves from animated height fields. ACM Trans. Graph. 32, 1, Article 2 (January 2013), 9 pages.
DOI = 10.1145/2421636.2421638
http://doi.acm.org/10.1145/2421636.2421638

1. INTRODUCTION

Physics-based animation in a standard visual effects pipeline is faced with some unique difficulties in terms of the balance between artistic control — what a shot is required to look like based on a nonphysical previsualization (previs) approved by the director — and physical realism — how the phenomenon would naturally evolve forward without control. In this article we tackle a common and important instance in the form of ocean waves.

Much attention has been devoted to the synthesis and evolution of ocean waves from physical principles, both in continuum mechanics [Lautrup 2005] and computer graphics [Tessendorf 1999; Bridson 2008; Darles et al. 2011]. However, as far as we know nobody has researched how to match physically based waves to an existing animation produced by arbitrary means. Indeed, in production it is common for previs artists to use any number of techniques such as blending, smoothing, procedural noise and deformers (in addition to more physical FFT synthesis) to achieve the timing and rough shapes needed to tell the story. The resulting animated surface geometry cannot be directly imported back into an FFT-based tool for developing into the final shot.

In practice, effects artists may instead generate a large random FFT ocean from scratch and manually search through it for areas which roughly match the previs input. The more complex the previs waves, the harder and more tedious this search becomes. Alternatively, higher-frequency displacement detail can be layered directly on the previs geometry, but estimating the propagation speeds for the detail to be visually consistent with the apparent physics of the previs waves is difficult, and adjustments of the large previs waves to appear more realistic (e.g., by flattening troughs and sharpening peaks) is even harder. We tackle the problem head on. We take previs input in the form of an animated height field (a grid topology mesh) without making any assumptions about how it was animated. Our method makes the following contributions.

(1) We optimize fitting of a Fourier-based model of wave propagation to previs model data over the duration of the animation, solving for wave amplitudes, phase shifts, and speeds.
(2) We resolve mesh self-intersections caused by adding physically based horizontal displacement using a new optimization

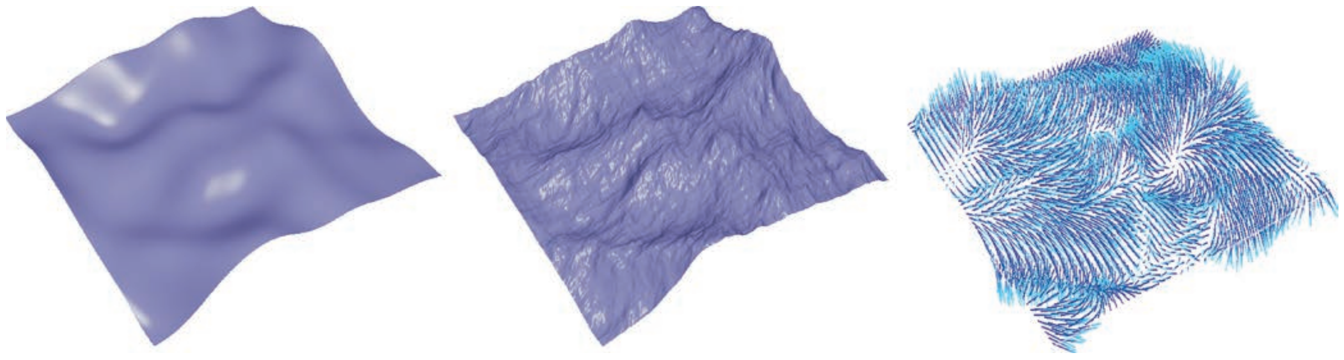


Fig. 1. (Left) Input previs of waves from a visual effects production environment. Only geometry is provided. Angular frequencies and amplitudes vary in time. (Middle) Output synthesized waves. Horizontal displacements and high frequencies have been added based on an estimation of wave parameters (amplitude, phase shift, and angular frequency) for each wave vector present in the input. This enhances the previs with flat troughs and sharp peaks that characterize true deep water waves while retaining a correspondence to and the timing of the previs. (Right) Analytical surface velocities projected onto a plane.

method which is significantly better at tracking the desired result than previous methods.

(3) We estimate physical depth required for the animation to drive a consistent dispersion relation for synthesizing higher-frequency detail with standard wave spectrum models.

(4) We optionally provide a physically consistent velocity (or time-varying displacement) field at and below the surface for additional processing.

Figure 2 illustrates these steps in cross-section. The first step in particular embodies the compromise we take between following the artistic input and following a physical model of waves. We generate waves as close as possible to the input animation that maintain coherent velocity and amplitude, a requirement which we argue broadly captures the physicality of deep ocean waves. However, we do not make any requirements on a particular dispersion relation (wave speed as a physically consistent function of wave length), and allow wave parameters to change over time, viewing this as a much less visible aspect of physics that could be modified for artistic reasons. If the input happens to be periodic and consistent with the physical model, the method reproduces those waves with high precision as can be seen in Figure 7. It should be mentioned that the linear physical model of waves, strictly speaking, is only valid for relatively small amplitudes compared to wave length. However, years of industry experience show that it can be pushed well beyond this limit and still produce convincingly natural animation. At the time of writing this article our new method is being evaluated in a large-scale production environment.

2. RELATED WORK

As general background, Lautrup [2005] provides a survey of waves in the context of continuum mechanics; Darles et al. [2011] survey techniques for ocean simulation in computer graphics.

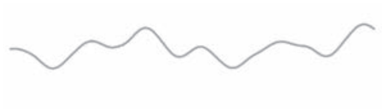
Ocean wave models in graphics begin with the Gerstner wave approaches of Peachey [1986] and Fournier and Reeves [1986]. Mastin et al. [1987] use the Fourier transform to synthesize plausible waves by filtering random coefficients to match the Pierson-Moskowitz spectrum. Tessendorf [1999] later presented Fourier-based methods for synthesizing, animating, and rendering realistic oceans which have since become standard across the industry. Tessendorf also discusses adding “chop” with horizontal displacements, and a simple approach to detect where this causes a self-intersection. Angelidis et al. [2011] mention a procedural method for removing self-intersections (but do not provide any detail) and develop a design tool for layout artists.

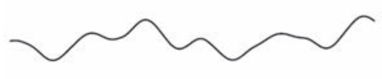
Many authors have worked in the general area of fluid control, matching input simulations/animations but not tackled ocean waves [Treuille et al. 2003; McNamara et al. 2004; Shi and Yu 2005a, 2005b; Hong and Kim 2004; Fattal and Lischinski 2004; Thürey et al. 2006; Nielsen et al. 2009; Nielsen and Christensen 2010; Rasmussen et al. 2004; Huang et al. 2011; Nielsen and Bridson 2011; Bhatacharya et al. 2012]. Mihalef et al. [2004] propose a method for designing and simulating breaking waves by lofting 2D wave slices which are then used as input to a full 3D simulation, but this does not fit the deep ocean previs pipeline.

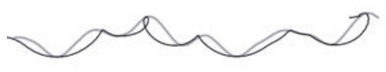
Other authors have explored using the forward Fourier transform for estimating and synthesizing ocean waves from an input sequence of some form. Thon and Ghazanfarpour [2002] propose a method for estimating adaptively sampled amplitudes and spatial frequencies from the FFT of a picture of waves. The amplitudes and spatial frequencies are based on a quadtree refinement of the image spectrum in Fourier space. The method considers only a single image and hence does not consider the temporal dimension: in particular the method does not estimate phase shifts and dispersion relations, and the waves are animated by shifting the phases of the estimated waves. Frechot [2007] similarly samples ocean spectrums based on a quadtree representation of wave vectors and amplitudes.

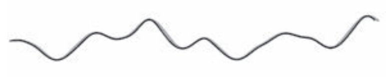
Tessendorf [1999] describes a technique used for estimating wave amplitudes and speeds based on a Power Spectral Density (PSD) analysis of a video sequence. The result of the PSD, which consists of Fourier transforms in time and space, is a numerical approximation of a statistical average of the amplitude as a function of spatial and angular frequency. This approach typically employs 1–2 minutes of video footage in order to obtain a full spectrum in both time and space. Our proposed estimation algorithm employs Fourier transforms in space combined with a nonlinear optimization in time. This allows us to use data from a relatively small window in time and detect temporal changes in dispersion under the assumption that angular frequency is a function of the wave vector and time.

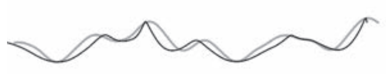
Spencer et al. [2006] determine the scale of waves and sea state from video. In particular the scale is found by assuming a deep water dispersion relation and measuring the difference between propagation speeds for different wavelengths. Sea state is estimated

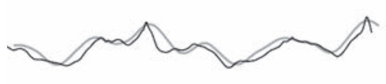

(a) input height field


(b) estimated waves


(c) estimated waves with x-displacements and self-intersections


(d) self-intersection handling based on Jacobian (approach is conservative and subject to temporal instability)


(e) our proposed self-intersection handling based on explicit optimization


(f) addition of synthesized high-frequency detail

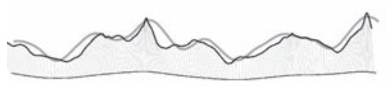

(g) shows an additional layer at finite depth with attenuated movement compared to the surface layer

Fig. 2. Illustration of the various steps involved in the estimation and synthesis of waves. Note that the signal in question is not periodic.

by fitting a two-parameter Phillips spectrum [Tessendorf 1999] (the unknowns being wind speed and Phillips amplitude) to the video sequence in PSD space. Given the wind speed, the sea state can be determined from a Beaufort chart. In contrast, we estimate dispersion relation, phase shift, and amplitude for all wave vectors.

3. ESTIMATION OF WAVE PARAMETERS

In this section we describe in detail our algorithm for estimating wave parameters from a time-varying input height field. For the applications considered in this article, the estimation of the wave parameters is important mainly for synthesizing the velocity field and adding higher-frequency detail that moves consistently with the coarse waves; knowledge of the exact wave parameters is not strictly required to add horizontal displacements for flatter troughs and sharper peaks.

ALGORITHM 1: $(A_{ij}(t_r), \theta_{ij}(t_r), \omega_{ij}(t_r))$ = estimateWaveParameters

```
Input: $h$ {input height field}
Input: $n_x, n_z, n_t$ {dimensions in space and time}
Input: $\tau$ {convergence threshold}
Input: $\Delta t$ {time-step between samples in time}
Input: $w$ {window of samples in time}
Require: $w \geq 3$
  for $r = 0 \to n_t$ do
    {solve for wave parameters at time $t_r$:}
    for $(i, j) = (-n_x/2+1, 0) \to (n_x/2-1, n_z/2-1)$ do
      {solve for Fourier modes $(i, j)$ and $(-i, -j)$:}
      {outer loop of non-linear least squares:}
      while residual $\geq \tau$ do
        non-linear search for $(\omega_{ij}, \omega_{-i,-j})$
        {linear least squares solve:}
        (residual, $B_{ij}$, $B_{-i,-j}$) = LSQ(Eq. (7),$\omega_{ij}$,$\omega_{-i,-j}$)
      end while
    end for
  end for
```

In the approach to synthesizing waves summarized by Tessendorf [1999], the wave parameters are all fixed in time. For matching previs input, however, it is necessary to allow these to vary with time and we perform the estimation as follows. Given a nonphysically based animated input height field sampled in space and time, $h(x_p, z_q, t_r)$, we wish to fit to h a sum of constant-frequency cosine waves. This cosine representation of waves is used in both graphics [Bridson 2008; Tessendorf 1999] and continuum mechanics [Lautrup 2005]. At each instant the cosine waves are estimated by keeping their phase-shift, amplitude, and angular frequency parameters constant inside a small temporally symmetric window. In particular the following expression is minimized at each point in time, t_r:

$$\sum_{s=-w/2}^{w/2} \sum_{p=0}^{n_x-1} \sum_{q=0}^{n_z-1} |h(x_p, z_q, t_{r+s}) - \tilde{h}(x_p, z_q, t_r, s)|^2 \tag{1}$$

where $x_p = pL_x/n_x$, $z_q = qL_z/n_z$, $t_r = rL_t/n_t$, $L_x \times L_z$ is the size of the domain in world space, $n_x \times n_z$ is the number of sample points, w is the window of samples in time contributing to the estimation at time t_r, and $\tilde{h}$ is the sum of cosine waves given by

$$\begin{aligned} \tilde{h}(x_p, z_q, t_r, s) \;=\; & \sum_{i=-n_x/2+1}^{n_x/2-1} \sum_{j=-n_z/2+1}^{n_z/2-1} A_{ij}(t_r) \\ & \cdot \cos\left(\vec{k} \cdot (x_p, z_q) - \Omega_{ij}^{rs} + \theta_{ij}(t_r)\right) \end{aligned} \tag{2}$$

where $\theta_{ij}(t_r)$ is the phase shift, $\Omega_{ij}^{rs} = \int_{t=0}^{t_r} \omega_{ij}(t)dt + s\Delta t \omega_{ij}(t_r)$ integrates the angular frequency, $\omega_{ij}(t_r)$ is the angular frequency, Δt is the time step used in the estimation, $\vec{k} = 2\pi(i/L_x, j/L_z)$ is the wave vector, $k = ||\vec{k}||_2$ is the wave number, and A_{ij} is the amplitude. The unknowns at each point in time t_r are the phase shifts $\theta_{ij}(t_r)$, the angular frequencies $\omega_{ij}(t_r)$, and the amplitudes $A_{ij}(t_r)$. The known quantity $\int_{t=0}^{t_r} \omega_{ij}(t)dt$ contained in Ω_{ij}^{rs} represents the phase change of the cosine wave solved for sequentially from the first to the previous frame and cannot in practice be coalesced with the unknown phase shift $\theta_{ij}(t_r)$. While the phase shift is mostly constant in time we observed that it is indeed important for the stability of the estimated waves to allow the phase shift to change over time. This is particularly the case when the amplitude of a certain frequency wave first decreases (to the extent where angular frequency estimation becomes unreliable) and later increases again.

In the preceding formulation of the minimization problem, each point sampled in space depends on all cosine waves, hence all unknowns are coupled. However, it turns out that by transforming the minimization problem into Fourier space, only the two cosine waves with spatial frequencies (i, j) and $(-i, -j)$ are coupled. This simplifies the minimization problem and allows us to solve for each such pair of spatial frequencies independently.

In particular, the discrete form of Parseval's theorem [Wong 2011] implies that $||\mathcal{F}[g]||_2^2 = n_x n_z ||g||_2^2$, where $\mathcal{F}[g]$ is the discrete Fourier transform of g. Hence minimizing Eq. (1) is identical to minimizing

$$\sum_{s=-w/2}^{w/2} \sum_{i=-n_x/2+1}^{n_x/2-1} \sum_{j=-n_z/2+1}^{n_z/2-1} |\mathcal{F}[h]_{ij}(t_r) - \mathcal{F}[\tilde{h}]_{ij}(t_r, s)|^2. \quad (3)$$

Since both h and $\tilde{h}$ are real-valued, their Fourier transform is complex conjugate even. The Fourier coefficients of a sum of cosine waves similar to Eq. (2) is given in Bridson [2008]. However, in our case the wave parameters depend on time and the angular frequency depends on direction of travel as well. The Fourier coefficients of the wave, $\tilde{h}$, in Eq. (2) are given by

$$\begin{aligned}\mathcal{F}[\tilde{h}]_{ij}(t_r) &= \frac{1}{2} e^{\sqrt{-1}(\theta_{ij}(t_r) - \Omega_{ij}^{rs})} A_{ij}(t_r) \quad (4)\\ &+ \frac{1}{2} e^{-\sqrt{-1}(\theta_{-i,-j}(t_r) - \Omega_{-i,-j}^{rs})} A_{-i,-j}(t_r).\end{aligned}$$

Hence, for each pair of spatial frequencies (i, j) and $(-i, -j)$ we minimize

$$\sum_{s=-w/2}^{w/2} |\mathcal{F}[h]_{ij}(t_r) - \mathcal{F}[\tilde{h}]_{ij}(t_r, s)|^2, \quad (5)$$

with respect to the six unknowns A_{ij}, $A_{-i,-j}$, θ_{ij}, $\theta_{-i,-j}$, ω_{ij} and $\omega_{-i,-j}$ at time t_r.

These equations exhibit a nonlinear dependence on the phase shifts and the angular frequencies. However, by rewriting the Fourier coefficients as

$$\begin{aligned}\mathcal{F}[\tilde{h}]_{ij}(t_r, s) &= \frac{1}{2} e^{-\sqrt{-1}(\Omega_{ij}^{rs})} B_{ij}(t_r) \quad (6)\\ &+ \frac{1}{2} e^{\sqrt{-1}(\Omega_{-i,-j}^{rs})} B_{-i,-j}(t_r)\end{aligned}$$

where $B_{ij} = e^{\sqrt{-1}(\theta_{ij}(t_r))} A_{ij}(t_r)$ and $B_{-i,-j} = e^{-\sqrt{-1}(\theta_{-i,-j}(t_r))} A_{-i,-j}(t_r)$ are complex numbers, we can reduce the nonlinear optimization to include only the angular frequencies. In particular, given B_{ij}, we have $A_{ij} = \sqrt{\mathrm{Re}(B_{ij})^2 + \mathrm{Im}(B_{ij})^2}$ and $\theta_{ij} = \tan^{-1}(\frac{\mathrm{Im}(B_{ij})}{\mathrm{Re}(B_{ij})})$.

Our estimation algorithm is summarized in Algorithm 1. At each consecutive point in time t_r and for each pair of spatial frequencies (i, j) and $(-i, -j)$, an outer loop performs a nonlinear optimization of ω_{ij} and $\omega_{-i,-j}$ over a window of time-samples w. At each iteration of the nonlinear optimization, we perform a linear least squares solve for B_{ij} and $B_{-i,-j}$, from which we can compute A_{ij}, $A_{-i,-j}$, θ_{ij} and $\theta_{-i,-j}$ as outlined before. Let

$$\mathcal{M} = \begin{bmatrix} \cos(\Omega_{ij}^{ra}) & \sin(\Omega_{ij}^{ra}) & \cos(\Omega_{-i,-j}^{ra}) & \sin(\Omega_{-i,-j}^{ra}) \\ -\sin(\Omega_{ij}^{ra}) & \cos(\Omega_{ij}^{ra}) & \sin(\Omega_{-i,-j}^{ra}) & -\cos(\Omega_{-i,-j}^{ra}) \\ \vdots & \vdots & \vdots & \vdots \\ \cos(\Omega_{ij}^{rb}) & \sin(\Omega_{ij}^{rb}) & \cos(\Omega_{-i,-j}^{rb}) & \sin(\Omega_{-i,-j}^{rb}) \\ -\sin(\Omega_{ij}^{rb}) & \cos(\Omega_{ij}^{rb}) & \sin(\Omega_{-i,-j}^{rb}) & -\cos(\Omega_{-i,-j}^{rb}) \end{bmatrix}$$

be the linear operator, where $a = -w/2$ and $b = w/2$. Then

$$\mathcal{M} \begin{bmatrix} \mathrm{Re}(B_{ij}) \\ \mathrm{Im}(B_{ij}) \\ \mathrm{Re}(B_{-i,-j}) \\ \mathrm{Im}(B_{-i,-j}) \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(\mathcal{F}[h]_{ij}(t_a)) \\ \mathrm{Im}(\mathcal{F}[h]_{ij}(t_a)) \\ \vdots \\ \mathrm{Re}(\mathcal{F}[h]_{ij}(t_b)) \\ \mathrm{Im}(\mathcal{F}[h]_{ij}(t_b)) \end{bmatrix}. \quad (7)$$

Ignoring degenerate situations, the linear system has full rank when the time window includes two samples. The full nonlinear equation system becomes well posed when the time window includes three samples and with more than three samples the system is overdetermined. In Section 5 we provide a more in-depth analysis of the effect of time step and window size on the error in estimated parameters. Two rows in the matrix are linearly dependent if $\Omega_{-i,-j}^{rs} = \{\pi - \Omega_{ij}^{rs}, 2\pi - \Omega_{ij}^{rs}\}$. Furthermore two odd- or even-numbered rows are linearly dependent if $\Omega_{-i,-j}^{rs_1} = \Omega_{-i,-j}^{rs_2} \pm \pi$ and $\Omega_{ij}^{rs_1} = \Omega_{ij}^{rs_2} \pm \pi$. Provided that this is not the case and both Ω_{ij}^{rs} and $\Omega_{-i,-j}^{rs}$ are nonzero (they can however be $\ll 1$), the aforesaid system has full rank. If one or both of Ω_{ij}^{rs} and $\Omega_{-i,-j}^{rs}$ are zero, a rank-2 system can be derived under the assumption that the wave traveling in either the positive or negative direction is identically zero.

The estimation of wave parameters is performed at the resolution of the input height field, whereas the output height (and velocity) fields are typically synthesized at higher resolution. The synthesis process is the subject of the next section.

4. SYNTHESIS OF HEIGHT AND VELOCITY

Given the estimated wave parameters for each wave vector, the low frequencies of the output height are synthesized by using Eq. (4) to construct a set of Fourier coefficients followed by an inverse FFT. Next a set of low-frequency horizontal displacements are computed in Fourier space, transformed to the spatial domain, and used to displace a regular lattice. The horizontal displacements will add sharper peaks and flatter troughs to the waves, but may introduce self-intersections (Figures 2(c) and 3(b)). These self-intersections are resolved in the spatial domain Figures 2(e) and 3(c). Finally the Fourier coefficients of the higher-frequency waves and their horizontal displacements are computed from a suitable spectrum, for example, the Phillips spectrum [Tessendorf 1999], as well as from knowledge of the angular frequencies of the estimated low-frequency waves. In this way the higher-frequency waves move consistently with the low-frequency input.

4.1 Adding Horizontal Displacements

The horizontal displacements can be synthesized using an inverse FFT. In particular the Fourier coefficients of the horizontal displacements are given by [Bridson 2008, formula (13.20)]: $(X_{ij}(t_r), Z_{ij}(t_r)) = \sqrt{-1}\frac{\vec{k}}{k} F[\tilde{h}]_{ij}(t_r)$. Note that the displacements do not depend explicitly on the estimated wave parameters, just

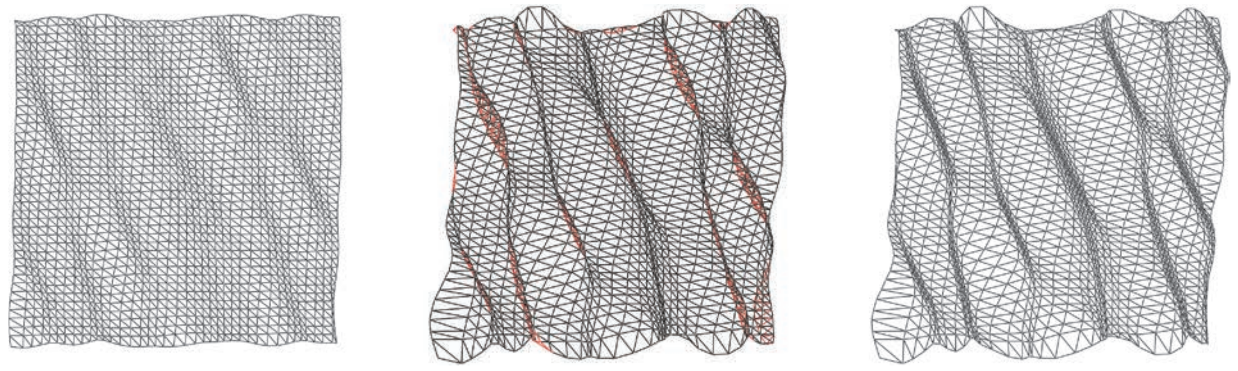

(a) self-intersection resolution by guaranteeing a positive Jacobian with a global displacement scale α is too conservative
(b) no self-intersection handling triangles with negative Jacobian are colored red
(c) self-intersection resolution using our proposed optimization on displacements

Fig. 3. Illustrates self-intersections occurring when adding horizontal displacements and the two main approaches explored for resolving these.

the Fourier coefficients. However, the Fourier coefficients of the synthesized higher-frequency waves depend on the estimated wave parameters and we need to compute horizontal displacements for these high-frequencies as well.

The horizontal coordinates in the spatial domain are computed as

$$x_{pq}(t_r) = \frac{p}{n_x} L_x + \alpha_{pq}(t_r)\Delta x_{pq}(t_r), \qquad (8)$$
$$z_{pq}(t_r) = \frac{q}{n_z} L_z + \alpha_{pq}(t_r)\Delta z_{pq}(t_r),$$

where Δx_{pq} and Δy_{pq} are the horizontal displacements computed by the inverse Fourier transform of $(X_{ij}(t_r), Z_{ij}(t_r))$, α_{pq} is the scale of the horizontal displacement at grid point (p, q) which is traditionally specified manually by an artist but can be computed automatically as described next. Setting $\alpha_{pq} = 1$ typically leads to self-intersections (Figures 2(c) and 3(b)). On the one hand we wish to avoid self-intersections, and on the other hand we wish to include as much of this horizontal displacement as possible because it adds realism (up to some limit) to both the shape and movement of the waves.

4.2 Resolving Self-Intersections

In this section we explore and discuss different methods for resolving self-intersections arising from horizontal displacements. We focus mainly on two approaches. The first approach guarantees a positive Jacobian of the horizontal displacements inspired by the work of Tessendorf [1999] (Figure 3(a)). The second approach is a novel algorithm that performs explicit optimization on the displacements directly (Figure 3(c)). The former approach is simple to implement and about an order of magnitude faster than the latter, but unfortunately it can be too conservative and suffer temporal instability. The latter approach is more robust and gives better results in general.

If an automatic approach was not a requirement, an option for resolving self-intersections would be to have an artist paint the scale of the displacements α_{pq} iteratively until the self-intersections were resolved, which obviously has down-sides in terms of tedious expert user involvement. Another automatic approach would be to procedurally remove the self-intersections by detecting lattice intersections and removing the loops seen in Figure 2(c), however the height of the output would no longer match the height of the input.

It has previously been observed [Fournier and Reeves 1986] that for a single trochoid, ensuring that $kA < 1$ will avoid self-intersections, where k is the wave number and A is the amplitude. However, this is only true for a single trochoid in isolation. When adding several trochoids, this property does not hold.

4.2.1 *Guaranteeing a Positive Jacobian.* Tessendorf [1999] proposes using the Jacobian of the displacements (see Tessendorf [1999] for an expression of the Jacobian) to detect areas of self-intersections in order to add foam and spray. Similarly the Jacobian can be used to resolve self-intersections since finding the maximum α_{pq} that guarantees a positive Jacobian everywhere will avoid self-intersections. Figure 2(d) illustrates the approach of finding a global α that guarantees a positive Jacobian everywhere. This approach is conservative and is subject to temporal instability (see the accompanying video). If we allow the α_{pq} to vary spatially and pose the problem of maximizing $\sum_{pq} |\alpha_{pq}|^2$ subject to a positive Jacobian, we obtain a quadratic optimization with linear constraints for a 1D curve and a quadratic optimization with quadratic constraints for a 2D surface. The problem can be simplified by assuming that the spatial derivative of α_{pq} is small compared to the spatial derivative of the displacements. In doing so, the problem will become a local equation for α_{pq} at each grid point. Unfortunately this approach does not work well in practice; the preceding assumption often turns out to be invalid and self-intersections remain. The computation of the Jacobian involves inverse FFTs to form the spatial derivatives of the displacements. To avoid these additional transforms and the quadratic constraints, we propose to perform an optimization on the displacements directly (as opposed to an optimization on the Jacobian). In addition this allows us to enforce constraints on the steepness of the waves. We describe this approach in detail next.

4.2.2 *Explicit Optimization on Displacements.* We pose the problem of computing a self-intersection-free lattice as a constrained optimization on the displacements. In particular we solve for the coordinates $(\tilde{x}_{pq}, \tilde{z}_{pq})$ free of self-intersections, by minimizing

$$\sum_{p=0}^{n_x-1} \sum_{q=0}^{n_z-1} |(x_{pq}, z_{pq}) - (\tilde{x}_{pq}, \tilde{z}_{pq})|^2 \qquad (9)$$

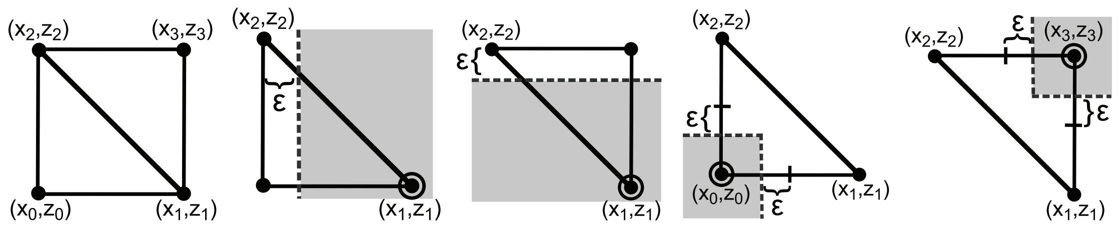


Fig. 4. Illustrates the constraints in Eq. (10) to Eq. (15) used to guarantee a self-intersection-free lattice in the optimization on displacements. The circled vertex is constrained to stay within the gray region.

subject to a set of linear constraints that guarantee a self-intersection-free lattice (defined shortly), where (x_{pq}, z_{pq}) are given by Eq. (8) with $\alpha_{pq} = 1$. The velocity field giving rise to the horizontal displacements is the gradient of a potential function since the waves are linearized [Lautrup 2005; Bridson 2008]. This means that the velocity field is irrotational so the horizontal deformations will not include global rotations, although the lattice may exhibit a significant amount of shearing. Due to this property the constraints do not have to take into account rotation and simple linear constraints suffice. Consequently the optimization falls into the class of quadratic programming. The linear constraints are formulated as six local inequality constraints per quad in the planar lattice. The constraints couple all vertices in the lattice due to the sharing of vertices between quads. Figure 4 depicts a quad and serves as an illustration of the constraints that ensure a self-intersection-free lattice. Next the constraints are given followed by an in-depth description.

$$\epsilon \leq x_1 - x_2 \tag{10}$$
$$\epsilon \leq z_2 - z_1 \tag{11}$$
$$\epsilon \leq 0.5(x_1 + x_2) - x_0 \tag{12}$$
$$\epsilon \leq 0.5(z_2 + z_1) - z_0 \tag{13}$$
$$\epsilon \leq x_3 - 0.5(x_1 + x_2) \tag{14}$$
$$\epsilon \leq z_3 - 0.5(z_2 + z_1) \tag{15}$$

Assuming that $\epsilon > 0$, Eq. (10) (Figure 4(b)) ensures that the point (x_1, z_1) stays in the half-plane $x \geq x_2 + \epsilon$. Similarly Eq. (11) (Figure 4(c)) guarantees that the point (x_1, z_1) stays in the half-plane $z \leq z_2 - \epsilon$. To avoid self-intersections we furthermore need to ensure that the point (x_0, z_0) does not cross the line segment $(x_1, z_1) \rightarrow (x_2, z_2)$ in the lower triangle (Eq. (12) and Eq. (13) in Figure 4(d)), and that the point (x_3, z_3) does not cross the line segment $(x_2, z_2) \rightarrow (x_1, z_1)$ in the lower triangle (Eq. (14) and Eq. (15) in Figure 4(e)). To ensure that the constraints remain linear in the latter two cases, the circled vertex is in each case restricted by the intersection of two half-planes.

The ϵ parameter can be used to adjust the minimum area of a deformed triangle and hence also the steepness of the resulting waves. For our results we use $\epsilon = \min(L_x/n_x, L_z/n_z)/4$. The quadratic programming problem can be solved within feasible time (from below a second to a couple of minutes) on a single thread for lattices of size 40×40 to 200×200. To avoid quadratic programming on too large lattices, we make the assumptions that the self-intersections are caused by the large-amplitude low frequencies present in the input height field and that the additional small-amplitude high-frequency detail added is free of self-intersections. This allows us to resolve the self-intersections at the resolution of the low-frequency input height field and next smoothly interpolate the self-intersection-free horizontal displacements to the resolution at which the output height field will be synthesized.

Figure 3(c) and the accompanying video show how the quadratic programming approach preserves a significant portion of the horizontal displacement (as opposed to the Jacobian-based approach in Figure 3(a)) while producing a lattice free of self-intersections.

4.3 Adding High-Frequency Waves

When adding higher-frequency waves, the speed at which they move is important to convey a more realistic wave motion. Part of what makes waves look realistic is that lower-frequency waves move faster than higher-frequency waves (see the accompanying video). In this section we address how to synthesize the higher-frequencies motivated by a physical approximation.

To add higher-frequency waves to the estimated low-frequency waves, the user specifies the wave number $k_{\min}$ starting from which high-frequency waves will be added. These higher-frequency waves can be added using any suitable spectrum: in this article we use the Phillips spectrum [Tessendorf 1999].

The angular frequency of the higher spatial frequency waves is computed based on an estimate of depth from the movement of the low spatial frequency waves. In particular the angular frequency (dispersion relation) is given by [Lautrup 2005]

$$\omega_k = \sqrt{gk \tanh(kd)}, \tag{16}$$

where g is the gravitational acceleration, d is the depth, and k is the wave number. Inverting this formula for depth we obtain $d = \tanh^{-1}(\omega_k^2/(gk))/k$. Since tanh converges asymptotically to one, the inversion $\tanh^{-1}$ is only reliable up to arguments of size approximately $1 - 10^{-6}$, depending on numerical accuracy. Arguments closer to one will return ∞. Hence for a reliable estimation of depth we must ensure $kd < \tanh^{-1}(1 - 10^{-6}) \approx 7.25$. If kd is larger, $\tanh^{-1}$ will return ∞ and the estimated depth d will be infinite as well. Since d is not known, the user has to settle on a maximum depth up to which a reliable estimate is desired and above which an infinite depth (deep water waves) can be accepted. Let the maximum reliable depth be given by $d_{\max}$, then we must ensure that for all wave numbers k used in the estimation of depth, $k < \min(\tanh^{-1}(1 - 10^{-6})/d_{\max}, k_{\min})$. To get a more robust depth estimate for the angular frequency computation we next combine the depths estimated for each reliable wave vector into a single scalar by setting $d = (\sum A_i d_i)/\sum A_i$, where A_i is the amplitude of the i'th contributing wave vector and d_i is the depth. Finally we use the estimated depth d in the dispersion relation formula Eq. (16) to compute the angular frequencies of the higher spatial frequency waves. This ensures that the higher spatial frequencies move consistently with the most dominating (in terms of amplitude) low spatial frequency waves.

4.4 Computing the Velocity Field Analytically

A volumetric velocity field extending below the surface can be computed by reconstructing the velocity at two layers (Figure 2(g)) followed by a computation of the potential flow using the two layers as a boundary condition [Nielsen and Bridson 2011]. The height, displacement, and velocity of the bottom layer are computed from the top layer by multiplying each frequency of these fields by e^{ky}, where k is the wave number and y is the (negative) depth. The volumetric velocity field can be used to drive particle simulations of bubbles, foam, and splashes or be used as input to a shape-based guiding method [Nielsen and Bridson 2011]. One of the advantages of using the estimated waves (as opposed to the original height field) as input to a shape-based guiding method is that the original height field will only have nonzero velocity in the vertical direction, whereas the estimation provides us with a full 3D velocity field (Figure 1(c)) The velocity field at a particular depth can be synthesized using an inverse FFT. The Fourier coefficients can be derived from Bridson [2008, formula (13.16)]. Furthermore we need to take into account that the angular frequency is time dependent and employ the fundamental theorem of calculus. We obtain the following expressions for the Fourier coefficients of the velocity field.

$$
\begin{aligned}
U_{ij}(t_r) &= \frac{1}{2}e^{\sqrt{-1}(\theta_{ij}-\Omega^{r0}_{ij})}\frac{A_{ij}(t_r)\omega_{ij}(t_r)2\pi i}{kL} \\
&\quad + \frac{1}{2}e^{-\sqrt{-1}(\theta_{-i,-j}-\Omega^{r0}_{-i,-j})}\frac{A_{-i,-j}(t_r)\omega_{-i,-j}(t_r)2\pi(-i)}{kL} \\
V_{ij}(t_r) &= \frac{1}{2}\sqrt{-1}e^{-\sqrt{-1}(\theta_{-i,-j}-\Omega^{r0}_{-i,-j})}A_{-i,-j}(t_r)\omega_{-i,-j}(t_r) \\
&\quad - \frac{1}{2}\sqrt{-1}e^{\sqrt{-1}(\theta_{ij}-\Omega^{r0}_{ij})}A_{ij}(t_r)\omega_{ij}(t_r) \\
W_{ij}(t_r) &= \frac{1}{2}e^{\sqrt{-1}(\theta_{ij}-\Omega^{r0}_{ij})}\frac{A_{ij}(t_r)\omega_{ij}(t_r)2\pi j}{kL} \\
&\quad + \frac{1}{2}e^{-\sqrt{-1}(\theta_{-i,-j}-\Omega^{r0}_{-i,-j})}\frac{A_{-i,-j}(t_r)\omega_{-i,-j}(t_r)2\pi(-j)}{kL}
\end{aligned}
$$

Note that these expressions are not simple functions of the Fourier coefficients for the height field, hence the estimated wave parameters are required to form the Fourier coefficients of the velocity field. Since the self-intersection handling changes the displacements we add a correction to the U_{ij} and W_{ij} components of the velocity field. In particular $U_{ij} \overset{+}{=} d(\tilde{x}_{pq} - x_{pq})/dt$ and $W_{ij} \overset{+}{=} d(\tilde{z}_{pq} - z_{pq})/dt$.

As an alternative to the analytical approach outlined before, the velocity field can be computed discretely using a finite difference approximation based on the displacements of the grid points of the lattice. The analytical approach produces a more accurate velocity field at the certain point in time it is evaluated, however it does not ensure that particles remain constrained on the surface when integrating over a single time step. A forward finite difference approach can ensure this.

5. RESULTS AND DISCUSSION

We have implemented our proposed algorithms in C++ and integrated them as a plugin in Maya. We use the Intel MKL for the FFT transforms as well as for the linear and nonlinear least squares. For solving the quadratic optimization problem we use OOQP [Gertz and Wright 2003]. All timings (seconds per frame) are listed in Table I and are obtained using a computer equipped with an 8 core 2.66 GHz Intel Xeon processor and 16GB of memory. OOQP is currently single-threaded so the reported timings for self-intersection resolution are for a single thread only. Recent work has explored parallel algorithms for quadratic programming and near-linear speedups in the number of processors have been reported [Gondzio and Grothey 2007].

Table I. Timings in Seconds per Frame (average)

example	production waves figure 1	noise waves figure 5	Phillips spectrum waves figure 6
estimation	0.37	0.031	0.0050
synthesis	0.86	0.87	0.86
self-intersection	0.63	1.1	1.3
total	1.9	2.0	2.1

The synthesis time includes the time for synthesizing both velocity and height. Estimation is done at resolution 50^2 and synthesis is done at resolution 2000^2.

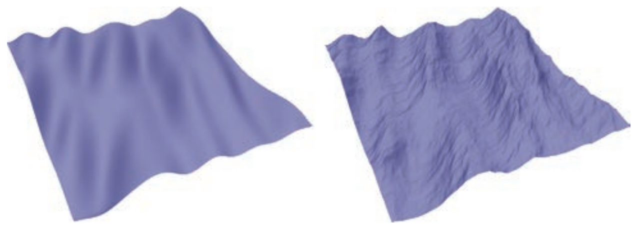


Fig. 5. (Left) Input waves formed by summing Perlin [1985] noise with varying amplitudes, wave vectors, and angular frequencies. (Right) Output waves with horizontal displacements and details.

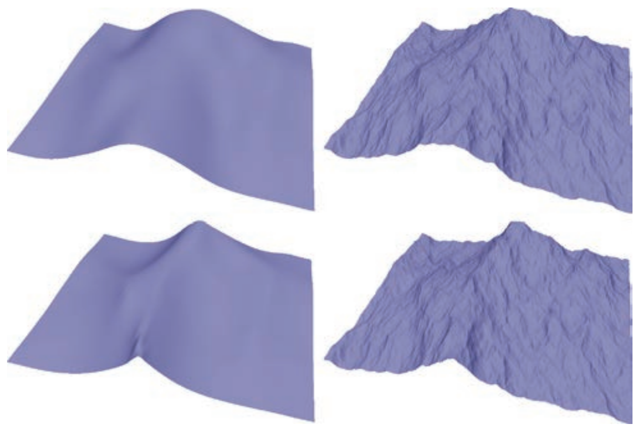


Fig. 6. (Left) Input waves formed by a summation of low-frequency waves based on the Phillips spectrum [Tessendorf 1999]. (Right) Output synthesized waves with horizontal displacements and high-frequency details added. Horizontal displacements have been added to the input surface in the bottom row, but not the top row. Notice how the output surface preserves the sharpness present in the input surface shown in the bottom row.

To evaluate the accuracy of our estimation algorithm we compute the maximum norm of the relative error in estimated amplitude, phase shift, and angular frequency compared to ground truth. The parameters of the input waves are temporally constant and initialized from a random variable with uniform distribution. In the test we used $n_t = 100$, $n_x = 20$, $n_z = 20$, $w = 6$ and the convergence threshold $\tau = 10^{-11}$. As shown in Figure 7, the maximum norm of the relative error over all wave vectors is below $6 \cdot 10^{-7}$ for amplitude, below $5 \cdot 10^{-4}$ for phase shift, and below $1 \cdot 10^{-5}$ for angular frequency.

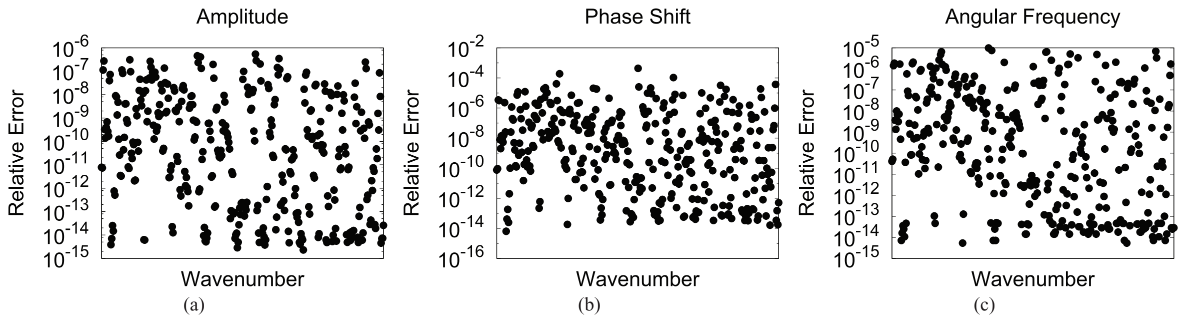


Fig. 7. Shows the maximum norm over 100 frames of the relative error in estimated amplitude, phase shift, and angular frequency on a logarithmic scale. The input amplitudes, phase shifts, and angular frequencies are temporally constant and are generated randomly with a uniform distribution for each wave vector.

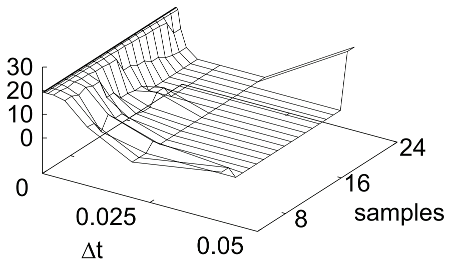


Fig. 8. Maximum norm of error in estimated parameters as a function of Δt and the number of samples in time.

Figure 8 depicts how the maximum norm in the error of estimated parameters varies as a function of the time step, Δt, and the number of samples in time for the test in Figure 7. In practice we use $\Delta t = 1/24$ of a second and six samples in time centered about the current time step and distributed evenly as integer multiples of Δt. This strikes a balance between faster performance (fewer samples in time included in the estimation) and a low error. As Figure 8 illustrates, the behavior of including additional samples in time is more complicated than a simple time-averaging of the estimated parameters as this would result in a steady increase in the residual. It can furthermore be observed that the residual does not tend to zero as $\Delta t \to 0$. We hypothesize that this is caused by numerical issues involving predominantly the slower moving waves requiring a certain Δt in order to reliably estimate the speed. We have not proven convexity of the nonlinear objective function, which would be implied by the Hessian of $\|b - \mathcal{M}(\mathcal{M}^T\mathcal{M})^{-1}\mathcal{M}^T b\|_2^2$ with respect to the variables $(\omega_{ij},\ \omega_{-i,-j})$ being positive semidefinite, where b is the right-hand side of Eq. (7) and $\mathcal{M}(\mathcal{M}^T\mathcal{M})^{-1}\mathcal{M}^T b$ is the solution to the normal equation of the least squares system in Eq. (7). However, in practice we have never observed any troubles with convergence or hints that multiple local minima exist: a single starting guess for the nonlinear estimation appears to suffice.

Figure 1 shows an example of previs waves originating from a visual effects production environment. The intent of the previs height field is to model a set of static sand dunes that gradually transition into waves in a stormy ocean. Both angular frequencies and amplitudes vary in time; see the accompanying video. The input previs is sampled and wave parameters estimated on a 50^2 grid and the output synthesized on a 2000^2 grid. This example is representative of the type of input an artist will input to the system in practice. The realism of the output waves is to a large extent limited by the input height field and illustrative of our choice to closely follow the input height field in order not to compromise the art direction, even though the wave dispersion changes in a nonphysical way over time. In Figure 5 we have constructed an input height field by summing Perlin noise with varying amplitudes, wave vectors, and angular frequencies. The wave parameters are estimated on a 50^2 grid and the output waves are synthesized on a 2000^2 grid. The accompanying video shows a comparison between the output Perlin noise animation with and without addition of horizontal displacements. The animation without horizontal displacements is representative of what an artist can do *manually* with existing tools, whereas our method is *automated* and allows for addition of horizontal displacements. Figure 6 depicts an example where the input consists of low frequencies of a Phillips spectrum. Our algorithm detects the deep water dispersion relation present in the input and uses this to synthesize the higher-frequency detail. The wave parameters of the input are estimated on a 50^2 grid and the output synthesized on a 2000^2 grid. The accompanying video shows the output Phillips spectrum animation with a buoy attached to a vertex in the deforming lattice. The high-frequency circular movement of the buoy is indicative of the high-frequency waves passing by underneath.

There are several limitations to our approach. Currently we do not explicitly handle nonperiodic signals. This means that the synthesized signal may in some cases deviate considerably from the input close to the boundary of the domain. Furthermore the self-intersection resolution algorithm performing explicit optimization on displacements does not guarantee the resulting lattice will be periodic even though the input height field is periodic. However, we emphasize that for most of our practical applications the input height field is itself not periodic and comprises the entire visual area of interest. Future work should address cases where the input is required to be tiled. The physically based formula used to compute the angular frequency from the depth (Eq. (16)) limits the *maximum* speed of the synthesized detail to be the speed of deep water waves. This means that if the input height field exhibits wave speeds above the speed of deep water waves, the high-frequency detail will currently not move fast enough. In practice we have so far not experienced this to be a problem. Also note that in reality capillary waves can move faster than deep water waves, but in this work we do not model waves in the capillary regime. Our method can also be used to detect dispersions in height fields originating from simulations of the shallow water and wave equations. Since our method works in the nonlocal Fourier domain, the depth can vary in time but is assumed spatially constant. Adding horizontal displacements will introduce sharper peaks and flatter troughs, but may change the horizontal positioning of the individual waves such that they do not perfectly line up with the input previs. It is left up to the individual artist to control the amount of horizontal displacements and thus how much the synthesized waves are allowed to deviate from the input. One could argue that resolving self-intersections is incorrect because a self-intersection signals that the amplitude has been pushed beyond the accuracy limits of the linear wave model. However, in practice artists will often set the amplitudes higher than what is physically plausible to obtain a desired dramatic effect.

6. CONCLUSION

In this article we proposed a set of automated algorithms for synthesizing waves from animated height fields. We demonstrated the ability of the algorithms to relatively closely match the input while adding sharper peaks, flatter troughs, and higher-frequency detail. Looking to the future we argue bringing physics into the previs stage, in an unobtrusive way, would be a significant help in avoiding difficult compromises where the previs input is inadvertently far from physically correct. We envisage accelerating our method to make an interactive tool where the previs artists could use all their existing techniques but also see how far from physically consistent their current animation is, or what depth or wave spectrum it implies, and adjust the animation towards a desired physical consistency. We also believe the estimation of wave parameters proposed in this article may be of use to analyze the effectiveness of a particular wave dampening approach [Söderström et al. 2010].

ACKNOWLEDGMENTS

We wish to thank the following individuals for their help and support of our work: Joe Letteri, Sebastian Sylwan, Kevin Romond, Christoph Sprenger, Dave Gouge, Natasha Turner as well as Diego Trazzi, Areito Echevarria, Brandon Davis and Tim Ebling, Mark Leone and Paul Roberts. In addition we wish to thank the referees for their insightful comments and suggestions.

REFERENCES

Angelidis, A., Anon, J., Bruins, G., and Reisch, J. 2011. Ocean mission on cars 2. In *ACM SIGGRAPH Talk,* Article No. 17.

Bhatacharya, H., Nielsen, M. B., and Bridson, R. 2012. Steady state stokes flow interpolation for fluid control. In *Proceedings of Eurographics '12 Short Paper.*

Bridson, R. 2008. *Fluid Simulation for Computer Graphics.* AK Peters.

Darles, E., Crespin, B., Ghazanfarpour, D., and Gonzato, J.-C. 2011. A survey of ocean simulation and rendering techniques in computer graphics. *Comput. Graph. Forum 30,* 1, 43–60.

Fattal, R. and Lischinski, D. 2004. Target-Driven smoke animation. In *Proceedings of ACM SIGGRAPH'04: Papers.* 441–448.

Fournier, A. and Reeves, W. T. 1986. A simple model of ocean waves. *SIGGRAPH Comput. Graph. 20,* 75–84.

Frechot, J. 2007. Realistic simulation of ocean surface using wave spectra. *J. Virt. Reality Broadcast. 4,* 11.

Gertz, E. M. and Wright, S. J. 2003. Object-Oriented software for quadratic programming. *ACM Trans. Math. Softw. 29,* 1, 58–81.

Gondzio, J. and Grothey, A. 2007. Parallel interior-point solver for structured quadratic programs: Application to financial planning problems. *Ann. Oper. Res. 152,* 319–339.

Hong, J.-M. and Kim, C.-H. 2004. Controlling fluid animation with geometric potential: Research articles. *Comput. Animat. Virtual Worlds 15,* 3–4, 147–157.

Huang, R., Melek, Z., and Keyser, J. 2011. Preview-Based sampling for controlling gaseous simulations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '11).* ACM, New York, 177–186.

Lautrup, B. 2005. *Physics of Continuous Matter.* IOP Publishing Ltd.

Mastin, G. A., Watterberg, P. A., and Mareda, J. F. 1987. Fourier synthesis of ocean scenes. *IEEE Comput. Graph. Appl. 7,* 16–23.

McNamara, A., Treuille, A., Popovic, Z., and Stam, J. 2004. Fluid control using the adjoint method. In *Proceedings of the ACM SIGGRAPH'04: Papers.* ACM, New York, 449–456.

Mihalef, V., Metaxas, D., and Sussman, M. 2004. Annimation and control of breaking waves. In *Proceedings of the ACM/Eurographics Symposium on Computer Animation.* 315–324.

Nielsen, M. B. and Bridson, R. 2011. Guide shapes for high resolution naturalistic liquid simulation. In *Proceedings of the ACM SIGGRAPH'11: Papers.* ACM, New York, 83:1–83:8.

Nielsen, M. B. and Christensen, B. B. 2010. Improved variational guiding of smoke animations. *Comput. Graph. Forum 29,* 2, 705–712.

Nielsen, M. B., Christensen, B. B., Zafar, N. B., Roble, D., and Museth, K. 2009. Guiding of smoke animations through variational couplings of simulations at different resolution. In *Proceedings of the ACM/Eurographics Symposium on Computer Animation.* 206–215.

Peachy, D. R. 1986. Modeling waves and surf. *SIGGRAPH Comput. Graph 20,* 65–74.

Perlin, K. 1985. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85).* ACM, New York, 287–296.

Rasmussen, N., Enright, D., Nguyen, D. Q., Marino, S., Sumner, N., et al. 2004. Directable photorealistic liquids. In *Proceedings of the ACM/Eurographics Symposium on Computer Animation.* 193–202.

Shi, L. and Yu, Y. 2005a. Controllable smoke animation with guiding objects. *ACM Trans. Graph. 24,* 1, 140–164.

Shi, L. and Yu, Y. 2005b. Taming liquids for rapidly changing targets. In *Proceedings of the ACM/Eurographics Symposium on Computer Animation.* 229–236.

Soderstrom, A., Karlsson, M., and Museth, K. 2010. A pml-based nonreflective boundary for free surface fluid animation. *ACM Trans. Graph. 29,* 136:1–136:17.

Spencer, L., Shah, M., and Guha, R. K. 2006. Determining scale and sea state from water video. *IEEE Trans. Image Process. 15,* 6, 1525–1535.

Tessendorf, J. 1999. Simulating ocean water. In *SIGGRAPH'99 Course Notes.*

Thon, S. and Ghazanfarpour, D. 2002. Ocean waves synthesis and animation using real world information. *Comput. Graph. 26,* 1, 99–108.

Thurey, N., Keiser, R., Pauly, M., and Rude, U. 2006. Detail-Preserving fluid control. In *Proceedings of the ACM/Eurographics Symposium on Computer Animation.* 7–12.

Treuille, A., McNamara, A., Popovic, Z., and Stam, J. 2003. Keyframe control of smoke simulations. In *Proceedings of the ACM SIGGRAPH'03: Papers.* 716–723.

Wong, M. W. 2011. *Discrete Fourier Analysis.* Springer.

Received November 2011; revised March 2012; accepted May 2012