Dispersion Kernels for Water Wave Simulation

José A. Canabal[1] David Miraut[1] Nils Thuerey[2] Theodore Kim[3,4] Javier Portilla[5]
Miguel A. Otaduy[1]

[1]URJC Madrid [2]Technical University of Munich [3]Pixar Animation Studios
[4]Universiy of California, Santa Barbara [5]Instituto de Optica, CSIC

Figure 1: *Rain on the pond. Our wave simulation method captures the gravity waves present in the wakes produced by the paper birds, as well as the capillary waves produced by rain drops, all with correct scale-dependent velocities. The waves reflect on all the objects in the scene, both static and dynamic. The domain is* 4 *meters wide, and is simulated on a* 1024×1024 *grid, at just* 1.6 *sec. per frame.*

Abstract

We propose a method to simulate the rich, scale-dependent dynamics of water waves. Our method preserves the dispersion properties of real waves, yet it supports interactions with obstacles and is computationally efficient. Fundamentally, it computes wave accelerations by way of applying a dispersion kernel as a spatially variant filter, which we are able to compute efficiently using two core technical contributions. First, we design novel, accurate, and compact pyramid kernels which compensate for low-frequency truncation errors. Second, we design a *shadowed convolution* operation that efficiently accounts for obstacle interactions by modulating the application of the dispersion kernel. We demonstrate a wide range of behaviors, which include capillary waves, gravity waves, and interactions with static and dynamic obstacles, all from within a single simulation.

Keywords: fluid simulation, water waves

Concepts: •**Computing methodologies** → **Physical simulation;**

1 Introduction

The dynamics of water waves obey a complex balance between the major forces acting on them, notably internal pressure, gravity, and surface tension. The Airy model [Airy 1849] describes this balance as a function of the depth of the liquid and the spatial frequency of the wave (a.k.a. wave number), and succeeds in capturing the characteristic dispersion of real-world waves.

Spectral approaches have been very successful in efficiently synthesizing height field waves with correct dispersion [Tessendorf 2004b]. However, handling obstacle boundaries in the frequency domain is computationally expensive, and quickly renders any approach in this direction impractical. Thus, users have to resort to localized three-dimensional simulations [Nielsen and Bridson 2011], precomputations [Jeschke and Wojtan 2015], or height field approaches with spatial filtering techniques [Kass and Miller 1990; Tessendorf 2008]. While the latter methods are typically fast, they only roughly approximate or even neglect dispersion effects.

This is the gap we aim to fill: our method allows for efficient, height field wave simulations in the spatial domain. The simulated waves correctly obey the Airy dispersion model, and our method can capture the nonlinearities of wave reflections at static and dynamic boundaries. Additionally, our method is fast enough to allow for interactive simulations on grids of moderate size.

In contrast to spectral approaches, we propose a spatial implementation of the Airy dispersion model. As shown in Section 3, this is done through convolution of the height field with a *dispersion kernel*. This choice is motivated by the fact that obstacle handling can then be introduced by spatially modulating the dispersion kernel.

Unfortunately, the spatial support of the naïve dispersion kernel is large, and this makes the approach *a priori* impractical. Moreover, we have observed that standard multiresolution approaches, which could enable a more compact kernel on each resolution, suffer unacceptable accuracy and robustness limitations. In Section 4, we introduce a novel method to design multiresolution kernels that are compact yet accurate, based on low-frequency compensation of truncation errors. Our strategy yields compact multiresolution kernels with imperceptible errors on wave dispersion.

Using the dispersion kernel as a reference, we incorporate reflecting boundary conditions with static and dynamic obstacles by modulating the kernel on areas occluded by obstacles. In Section 5, we introduce a *shadowed convolution* operation that approximates reflecting boundary conditions in a way that balances accuracy and computational cost. We propose a massively parallel algorithm for the computation of obstacle shadow masks, which introduces only a small overhead on the regular convolution operation.

We show results with wave dynamics at different scales, from capillary to gravity waves (see the pond scene in Fig. 1), all simulated

SA '16 Technical Papers,, December 05-08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982415

ACM Reference Format
Canabal, J., Miraut, D., Thuerey, N., Kim, T., Portilla, J., Otaduy, M. 2016. Dispersion Kernels for Water Wave Simulation. ACM Trans. Graph. 35, 6, Article 202 (November 2016), 10 pages.
DOI = 10.1145/2980179.2982415 http://doi.acm.org/10.1145/2980179.2982415.

Figure 2: *A* 10*-meter-wide and* 1*-meter-deep domain is perturbed, and we compare the resulting waves with our method (left, on a* 256×256 *2D grid), a 3D level set simulation [Thuerey and Pfaff 2016] (middle, on a* $256 \times 256 \times 128$ *3D grid), and the* iWave *method [Tessendorf 2004a] (right, also on a* 256×256 *2D grid). All images show the simulation after* 6 *seconds. The difference in wavefront propagation velocity between our method and the level set solution is only* 4.1%*. However, our method runs at* 30 *fps and the level set solution runs at* 20 *seconds/frame (although it could be further optimized). Moreover, due to numerical dissipation, the level set solution misses the high-frequency waves, which are in turn captured with our method.* iWave *considerably underestimates the propagation velocity.*

under one method. Waves reflect both on static and dynamic objects, producing rich interactions. On grids of up to 1024×1024 cells, our simulations run at 0.6 fps for the small-scale pond scene, and 5 fps for the large-scale canal scene in Fig. 13 (for $1/60$ sec. frames).

We have also validated that our method produces wave dispersion behavior that is extremely similar to a 3D level set simulation, but with even higher frequency content, and at just a small fraction of the cost, as shown in Fig. 2.

2 Related Work

The Navier-Stokes equations are the most commonly used physical model for fluids. In computer graphics, Foster and Metaxas [1996] were the first to solve them in three dimensions for animation purposes. The *stable fluids* algorithm, introduced by Stam [1999], is widely used in different forms, and has seen numerous improvements over the years, such as surface tracking for liquids [Foster and Fedkiw 2001], the *ghost-fluid* method for improved interface motions [Enright et al. 2003], and guiding techniques for artistic control [Nielsen and Bridson 2011]. Thorough overviews of the different algorithms for grid-based and particle-based methods can be found in corresponding survey texts [Bridson 2015; Ihmsen et al. 2014]. While a three-dimensional solver with surface tension forces could theoretically resolve the correct motion of gravity and capillary waves, the volumetric representation and correspondingly high computational cost make this approach infeasible in most practical cases.

Surface waves that do not break are well represented by height fields, which typically significantly reduce the computational cost due to their two-dimensional nature. The simplest model here is the linear wave equation, which is commonly used for graphical effects [Kass and Miller 1990]. It can be solved very efficiently, but leads to a single propagation speed for all waves. Closely related are shallow-water models [Wang et al. 2007; Chentanez and Müller 2010], which typically include transport and ground effects. To circumvent the single speed approximation, methods such as *iWave* [Tessendorf 2004a], and Bi-Laplacian surface energy formulations [Yu et al. 2012] were introduced. While these methods can compute a sub-class of dispersive motions, the dispersion is fixed, and cannot yield the correct physical range of wave motions. The *iWave* kernel was designed to be scale-independent. In essence, kernel values are independent of spatial resolution, and while the method produces dispersion within the range of wave numbers represented by the simulation, the dispersion law does not incorporate the scale and resolution of the domain. As shown in Fig. 2, we have observed that the propagation speed with *iWave* is often considerably lower than expected. A common theme of the height field methods mentioned so far is that they work in the spatial domain, and can flexibly handle arbitrary obstacle geometries. Similarly, wave particles [Yuksel et al. 2007; Cords 2008] and SPH-based shallow water particles [Solenthaler et al. 2011] represent waves with Lagrangian particles that can be updated very efficiently.

Due to their efficiency, height field methods are attractive in combination with 3D solvers, e.g., to capture detailed wave effects for splashing liquids [Patel et al. 2009], capillary waves with droplet interactions [Thürey et al. 2010], or surface turbulence [Kim et al. 2013; Mercier et al. 2015]. While it is an interesting direction to combine 2D and 3D methods, we will leave such coupled approaches for future work, and focus on 2D simulations in the following.

In contrast to the methods above, it can be advantageous to simulate waves in the frequency domain. The spectral approach by Tessendorf [Tessendorf 2004b; Darles et al. 2011] is widely used to efficiently simulate large-scale ocean waves. Here, the frequency representation allows for a fine control of the wave spectrum and its motion. The large number of movies employing this technique highlight the importance of surface waves for visual effects [Thacker 2010]. While Nielsen et al. [2013] proposed a method to synthesize given shapes with waves for artist control, wave interactions with obstacles are typically very difficult to incorporate into spectral solvers. This motivates our choice to develop a method that performs spatial convolutions, so that waves in arbitrary, changing geometries can be simulated. A different approach for precomputing the wave propagation with complex boundaries was recently developed [Jeschke and Wojtan 2015]. While this algorithm allows for efficient waves with dispersion effects, it requires costly precomputations whenever the geometry of boundaries changes.

Other works have attempted some sort of spatial solution of dispersive waves before ours. Loviscach [2002] derived a convolution kernel that models the propagation of waves in time and space. This kernel occupies *a priori* the complete domain, and Loviscach designed a combination of spatial and frequency truncation to partially limit the computational cost. In addition to truncation errors that we will later on demonstrate, this additionally required a modified dispersion relationship, further distorting the correct wave motions. Ottosson [2011] started instead from the same dispersion kernel as we do, which computes wave accelerations by way of a convolution operation. To reduce the cost of the complete-domain kernel, he built a pyramid of kernels for different bands, and approximated each level of the pyramid using a sum of separable Gaussian filters. Although very efficient, the approach suffers from notable errors that can significantly affect wave motion. In addition

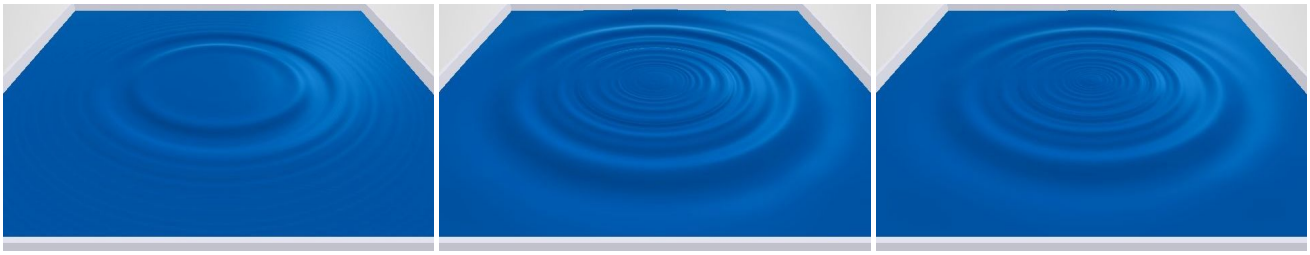


Figure 3: *Wave dynamics largely depend on their scale. From left to right, three examples of water wave evolution after a central perturbation: (i) A* 0.5*-meter-wide,* 0.2*-meter-deep domain, with* 1*-mm resolution,* 0.6 *seconds after the perturbation; (ii) a* 5*-meter-wide,* 1*-meter-deep domain, with* 10*-mm resolution, after* 3.5 *seconds; (iii) A* 50*-meter-wide,* 10*-meter-deep domain, with* 100*-mm resolution, after* 11 *seconds.*

to the severe truncation errors, a central limitation of the works of Loviscach and Ottosson is that neither of them handles wave reflections at obstacles. This means that, in practice, these approaches do not leverage the central advantage of spatial representations, while suffering an increased error in the wave propagation.

3 Wave Dispersion Kernel

In this section, we introduce relevant notation, we recall the dispersion relation under Airy wave theory, and we describe the computation of wave acceleration as a convolution operation with a dispersion kernel.

Let us define a generic wave through its frequency decomposition. For every position $\mathbf{x} \in \mathbb{R}^2$ and time t, the height of the wave may be expressed in general terms as:

$$h(\mathbf{x}, t) = \int_{\mathbb{R}^2} a(\mathbf{k})\, \mathrm{e}^{\mathrm{i}(\mathbf{k}^T \mathbf{x} - \omega(k)\, t + \phi(\mathbf{k}))} \mathbf{dk}, \tag{1}$$

where $\mathbf{k}$ is the wave vector of each frequency component, $k = \|\mathbf{k}\|$ is its wave number, $a(\mathbf{k})$ and $\phi(\mathbf{k})$ are the amplitude and phase of each frequency component, and $\omega(k)$ is its angular velocity.

Airy wave theory defines a linear model for the propagation of waves on the surface of a homogeneous fluid with uniform depth [Airy 1849; Dean and Dalrymple 1991; Lamb 1994]. The dispersion relation defines the propagation velocity corresponding to each wave number as:

$$c = \sqrt{\left(\frac{g}{k} + \frac{\sigma}{\rho}\, k\right) \tanh(k\, d)}, \tag{2}$$

where g is the gravity acceleration, ρ and σ are, respectively, the density and surface tension of the fluid, and d is the fluid depth. From the dispersion relation, we can derive that long waves (i.e., low wave number) are dominated by gravity (i.e., gravity waves). They propagate faster on deeper water and as wave length grows. Short waves (i.e., high wave number) are dominated by surface tension (i.e., capillary waves) and they travel faster as wave length decreases. Fig. 3 shows wave behaviors at three different scales. In the wakes shown in Fig. 1, wave fronts are dispersed into fast short capillary waves and slow longer waves.

In the remainder of the document, we drop the explicit time dependency of the wave height field. In addition, we denote with small letters functions defined in the spatial domain, e.g., $h(\mathbf{x})$, and with capital letters functions defined in frequency domain, e.g., $H(\mathbf{k}) = \mathcal{F}(h(\mathbf{x}))$, where $\mathcal{F}$ represents the Fourier transform.

The wave equation allows the computation of wave acceleration $\ddot{h}$ as a function of wave height:

$$\ddot{h}(\mathbf{x}) = c^2(k)\, \nabla^2 h(\mathbf{x}). \tag{3}$$

However, this equation is not of much practical use. The dispersion relation (2) defines for $c(k)$ a dependency w.r.t. wave number, but $h(\mathbf{x})$ is expressed in the spatial domain, without an explicit spectral decomposition. As a result, it is not possible to apply the appropriate propagation velocity to each wave number separately.

Transforming the equation to the frequency domain, through the application of the Fourier transform, we obtain instead a practical model:

$$\ddot{H}(\mathbf{k}) = -k^2\, c^2(k)\, H(\mathbf{k}) = F(\mathbf{k})\, H(\mathbf{k}). \tag{4}$$

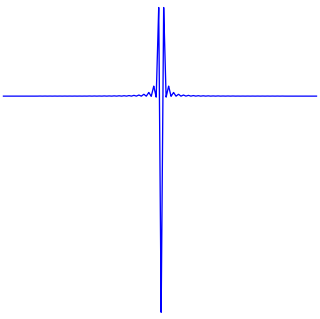
The Laplacian turns into a pointwise operation in frequency domain, thus removing the troublesome mix of spatial and frequency dependencies in the wave equation (3). As a result, each frequency component of the wave can be evolved in time independently, and its acceleration is given by the product of its height and the function $F(\mathbf{k}) = -k^2\, c^2(k) = -\omega^2(k)$. This property sets the basis for spectral wave methods [Tessendorf 2004b].

Transforming the wave acceleration function back to the spatial domain, through the inverse Fourier transform, we see that the wave acceleration can be computed through a convolution operation in the spatial domain:

$$\ddot{h}(\mathbf{x}) = f(\mathbf{x}) * h(\mathbf{x}). \tag{5}$$

The operator $*$ denotes convolution.

We refer to $f(\mathbf{x}) = \mathcal{F}^{-1}(-\omega^2(k))$ as the *dispersion kernel*, which satisfies the linear wave dispersion relation. The figure on the side shows an example dispersion kernel in 1D. At first sight, it appears to be well suited for compact truncation; however, we have observed that aggressive truncation produces significant error and quickly makes simulations unstable. Previous approaches have failed at producing compact yet accurate approximations of the kernel [Loviscach 2002; Ottosson 2011], making the convolution form of the wave equation (5) a mere theoretical result, without much practical use so far. The frequency form of the wave equation (4) turns out to be more computationally efficient and is used in practice [Tessendorf 2004b].



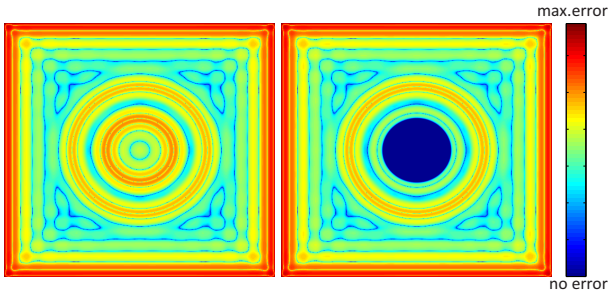


Figure 4: *We split a* 512×512 *kernel into high-frequency and low-frequency bands, and truncate the high-frequency band down to* 32×32 *in the spatial domain. The images compare kernel error (in frequency space) after adding the low-frequency band and the truncated high-frequency band. On the left, default error; on the right, error after our low-frequency compensation.*

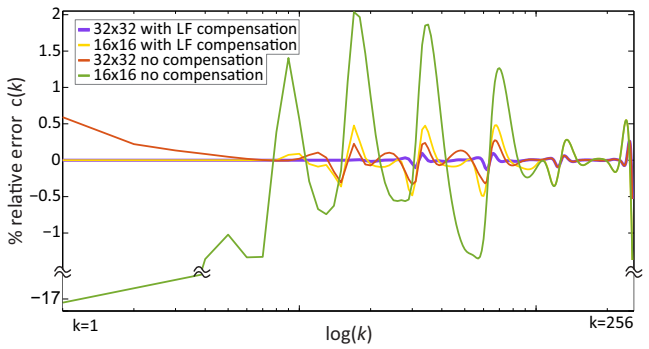


Figure 5: *We build kernel pyramids for a* 512×512 *kernel with different spatial truncation widths, with and without our low-frequency compensation. Then, we reconstruct the full kernel in each case by adding together all pyramid levels with appropriate low-pass filtering and upsampling, and compute the effective propagation velocity* c *for each wave number* k*. The plot compares the relative error in the effective wave dispersion* $c(k)$ *vs. the ideal solution (2). With* 32×32 *truncation and our low-frequency compensation, error is negligible. We use these settings in our simulations.*

However, it makes a strong inherent assumption on the spatial invariance of the dispersion kernel, hence it is used only for open-water phenomena with no complex obstacles.

Through the relation between wave number k, angular velocity ω, and propagation velocity $c = \omega/k$, we can also write the dispersion relation (2) in a convenient way for the evaluation of the dispersion kernel:

$$\omega^2 = \left(g\,k + \frac{\sigma}{\rho}\,k^3 \right) \tanh(k\,d). \tag{6}$$

Next, we introduce a compact yet accurate approximation of the dispersion kernel, which enables computationally efficient application of the convolution form of the wave equation. In addition, the use of the spatial domain formulation enables the treatment of complex obstacle interactions by modulating the dispersion kernel as a spatially variant filter.

4 Compact Pyramid Kernel Design

The advantages of the convolution form of the wave equation for obstacle treatment are easily surpassed by the computational cost induced by the large support of the dispersion kernel. We observe, however, that the reconstruction of higher frequencies requires a smaller support. Motivated by this observation, designing a multiresolution representation appears as a natural way to obtain a compact dispersion kernel, and Laplacian pyramids [Burt and Adelson 1983], or similar variants, offer a standard framework for this purpose. Unfortunately, truncation of the pyramid representation produces devastating errors in the computation of wave accelerations.

We start this section by analyzing the resulting errors in the application of a pyramid framework to the design of a multiresolution dispersion kernel. Then, we introduce our approach for the design of a compact yet accurate pyramid kernel, based on low-frequency compensation of truncation error. For a pyramid with N levels, the pyramid kernel consists of the full dispersion kernel at the coarsest resolution $f^{(N)}(\mathbf{x})$, and high-frequency dispersion kernels at all other resolutions $\{f_H^{(1)}(\mathbf{x}) \dots f_H^{(N-1)}(\mathbf{x})\}$. We conclude this section by outlining the complete multiresolution computation of wave accelerations.

4.1 Pyramid Kernel

The construction of a compact multiresolution dispersion kernel using pyramids works as follows. At each resolution, the dispersion kernel is separated into high-frequency and low-frequency parts. The high-frequency part is truncated to produce a compact kernel, while the low-frequency part is downsampled and the process is applied recursively. The runtime computation of acceleration works as follows. At each level of the pyramid, the corresponding high-frequency kernel is convolved with a downsampled version of the full height field, to produce the accelerations in a frequency band. At the lowest resolution, once the domain is small, the corresponding full kernel is applied instead. Finally, all the acceleration bands are progressively upsampled and added together.

For our analysis, we will consider one level of the pyramid, with frequencies smaller than $k_{\max}$, and with input dispersion kernel $F(\mathbf{k})$. We start by designing a high-pass filter $W_H(\mathbf{k})$ that will guide the separation into high- and low-frequency bands. For this purpose, we have used a cosine filter [Simoncelli and Freeman 1995]:

$$W_H(\mathbf{k}) = \begin{cases} \cos\left(\frac{\pi}{2}\,\log_2\left(\frac{2\,k}{k_{\max}}\right)\right), & \frac{k_{\max}}{4} < k < \frac{k_{\max}}{2} \\ 1, & k \geq \frac{k_{\max}}{2} \\ 0, & k \leq \frac{k_{\max}}{4} \end{cases} \tag{7}$$

The high-frequency dispersion kernel is then $W_H(\mathbf{k})\,F(\mathbf{k})$. Next, to produce a compact kernel, we smoothly truncate the high-frequency dispersion kernel in the spatial domain. For this purpose, we have used a raised cosine window with roll-off factor $\frac{1}{2}$:

$$s(\mathbf{x}) = \begin{cases} \frac{1}{2} + \frac{1}{2}\,\cos\left(\frac{3\,\pi\,M}{2\,R\,D}\,\|\mathbf{x}\| - \frac{\pi}{2}\right), & \frac{R\,D}{3\,M} < \|\mathbf{x}\| < \frac{R\,D}{M} \\ 1, & \|\mathbf{x}\| \leq \frac{R\,D}{3\,M} \\ 0, & \|\mathbf{x}\| \geq \frac{R\,D}{M} \end{cases} \tag{8}$$

D is the width of the domain, M the size of the domain in discrete samples, and R the desired radius of the kernel in samples.

After truncation, the resulting high-frequency dispersion kernel in the spatial domain is:

$$f_H(\mathbf{x}) = s(\mathbf{x})\,\mathcal{F}^{-1}\left(W_H(\mathbf{k})\,F(\mathbf{k})\right). \tag{9}$$

Similarly, the frequency-domain representation of the truncated high-frequency dispersion kernel is:

$$F_H(\mathbf{k}) = S(\mathbf{k}) * \left(W_H(\mathbf{k})\,F(\mathbf{k})\right). \tag{10}$$

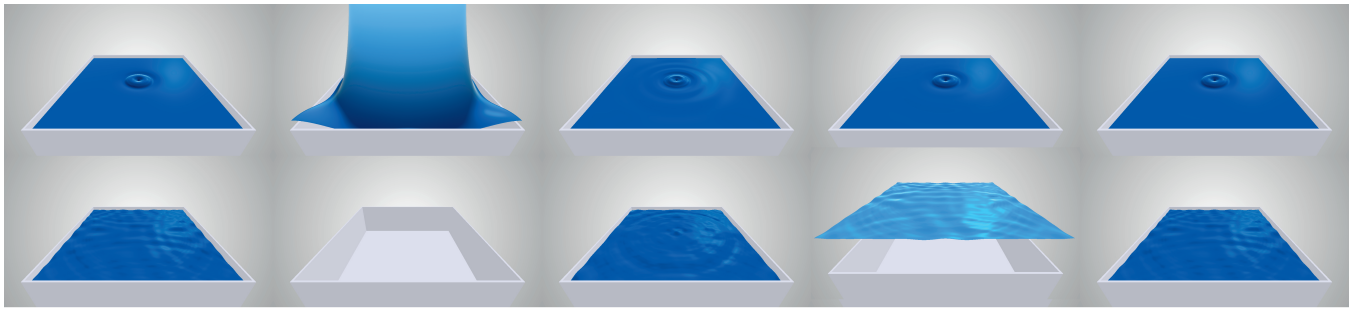


Figure 6: *The images compare two frames of a wave simulation on a* 128×128 *domain under different pyramid construction settings. With default spatial truncation, low-frequency error leads to volume change and low-frequency oscillations that turn into instability over time. With our low-frequency compensation, even aggressive* 12×12 *truncation yields plausible results, and with* 32×32 *truncation the error is imperceptible even after* 5 *seconds.*

As anticipated, truncation produces error. It produces high-frequency error, but it also spills notable error into low frequencies. This low-frequency error, evident in Fig. 4-left, produces volume change and low-frequency oscillations that turn into instability over time. Fig. 6 compares simulations with the full dispersion kernel, a pyramid implementation with truncated kernels, and our proposed solution to be described next.

We have also evaluated the approximate dispersion relations that result from truncating kernels at different sizes. Fig. 5 compares these dispersion relations to the accurate relation in (2).

4.2 Low-Frequency Compensation

The key insight behind our compact yet accurate pyramid kernel design is to fully compensate for the low-frequency error introduced by high-frequency truncation. We add this error to the standard low-frequency band of the kernel, and downsample the result to the next pyramid level. In this way, the low-frequency error of high-frequency truncation is compensated on the next pyramid level. The procedure is applied recursively.

Let us define as $E(\mathbf{k})$ the high-frequency truncation error:

$$E(\mathbf{k}) = F_H(\mathbf{k}) - W_H(\mathbf{k})\, F(\mathbf{k}). \tag{11}$$

We design the low-pass dispersion kernel by subtracting this error from the full dispersion kernel prior to low-pass filtering. In addition, we design the low-pass filter such that the reconstruction of the acceleration suffers only high-frequency error. Let us denote the (unknown) low-pass filter as $W_L(\mathbf{k})$. Then, the low-frequency dispersion kernel is defined as:

$$F_L(\mathbf{k}) = W_L(\mathbf{k})\,(F(\mathbf{k}) - E(\mathbf{k})). \tag{12}$$

The approximate wave acceleration is obtained by summing the contributions of the high- and low-frequency dispersion kernels. Note also that the low-frequency contribution is computed on the next pyramid level, and correct reconstruction requires low-pass filtering of the height field prior to downsampling, as well as low-pass filtering of the result of convolution prior to upsampling. We use the same low-pass filter $W_L(\mathbf{k})$ for these two operations. Then, by omitting downsampling and upsampling operations thanks to perfect reconstruction, the approximate wave acceleration $\ddot{H}^*(\mathbf{k})$ is computed as:

$$\ddot{H}^*(\mathbf{k}) = F_H(\mathbf{k})\, H(\mathbf{k}) + W_L(\mathbf{k})\ (F_L(\mathbf{k})\ (W_L(\mathbf{k})\, H(\mathbf{k}))). \tag{13}$$

By substituting the expressions for the truncation error (11) and the low-frequency kernel (12), and reordering terms, we obtain an expression of the approximate wave acceleration formulated as the sum of the accurate solution and an error term:

$$\ddot{H}^*(\mathbf{k}) = \underbrace{\left(W_H(\mathbf{k}) + W_L^3(\mathbf{k})\right)\ F(\mathbf{k})\, H(\mathbf{k})}_{\text{true acceleration } \ddot{H}(\mathbf{k})} + \underbrace{\left(1 - W_L^3(\mathbf{k})\right)\ E(\mathbf{k})\, H(\mathbf{k})}_{\text{error}}. \tag{14}$$

We design the low-pass filter $W_L(\mathbf{k})$ to match the true acceleration in the right term:

$$W_H(\mathbf{k}) + W_L^3(\mathbf{k}) = 1 \Rightarrow W_L(\mathbf{k}) = (1 - W_H(\mathbf{k}))^{\frac{1}{3}}. \tag{15}$$

And the resulting approximate acceleration is:

$$\ddot{H}^*(\mathbf{k}) = \ddot{H}(\mathbf{k}) + W_H(\mathbf{k})\, E(\mathbf{k})\, H(\mathbf{k}). \tag{16}$$

The resulting error term, $W_H(\mathbf{k})\, E(\mathbf{k})\, H(\mathbf{k})$, includes the high-pass filter $W_H(\mathbf{k})$; therefore, it exhibits only high-frequency error introduced by the high-frequency truncation. In Fig. 4, we show the error after reconstructing the full kernel by adding the truncated high-frequency kernel and the low-frequency kernel, with and without low-frequency compensation. With our kernel design, we achieve full low-frequency compensation of the truncation error.

For the construction of the full pyramid kernel, the full dispersion kernel $F^{(i)}(\mathbf{k})$ at level i is initialized as an ideally shrunk version of the low-frequency dispersion kernel $F^{(i-1)}(\mathbf{k})$ from the previous level. Finally, we apply the inverse Fourier transform to obtain the spatial representations of the kernels.

In Fig. 5, we compare the approximation of the dispersion relation with and without our low-frequency compensation, for different kernel sizes. Briefly, the RMS of relative error in wave propagation speed for a 32×32 pyramid kernel with low-frequency compensation is just 0.06%, and there is only numerical low-frequency error. For a pyramid kernel with no low-frequency compensation, the RMS relative error grows to 0.11%, but there is a 0.6% low-frequency error. Note that for a naïve 32×32 truncation of the full kernel (with a smooth window), not shown on the plot, the RMS relative error is 11.2% and the low-frequency error is 160%. For the same experiment, a 31×31 *iWave* kernel suffers an RMS relative error in wave propagation speed of 98%, with low-frequency error of 80%.

Fig. 6 illustrates how the truncation errors influence simulation results. Without compensation, the low-frequency error introduced by truncation leads to volume change and low-frequency oscillations that turn unstable over time. Instead, we obtain very high accuracy on the dispersion relation and the resulting simulations, under kernel sizes that are computationally affordable. In practice, we have used a kernel with radius $R = 16$ (i.e., 32×32) in all our examples. Further optimization of kernel shape and size would be possible through optimal design of the high-pass filter $W_H(\mathbf{k})$ in (7) and the truncation window $s(\mathbf{x})$ in (8). We leave these options for future work.

All our analysis and design assumes that the application of the dispersion kernel is a linear and spatially invariant operation, thus the possibility to transform computations to and from frequency and spatial domains. In Section 5 we will break the assumption, making the application of the dispersion kernel a spatially variant operation. This will introduce some small error in our computations. We will analyze and discuss this error in Section 6, and show that it is negligible in practical settings.

Algorithm 1 Simulation step

1: {Compute the height pyramid}
2: $H^{(1)}(\mathbf{k}) = \mathcal{F}(h^{(1)}(\mathbf{x}))$
3: **for each** level $i = 2$ **to** N **do**
4: $\quad H^{(i)}(\mathbf{k}) = \text{downsample}(W_L^{(i-1)}(\mathbf{k})\, H^{(i-1)}(\mathbf{k}))$
5: **end for**
6: **for each** level $i = 2$ **to** N **do**
7: $\quad h^{(i)}(\mathbf{x}) = \mathcal{F}^{-1}(H^{(i)}(\mathbf{k}))$
8: **end for**
9: {Compute the acceleration pyramid}
10: **for each** level $i = 1$ **to** $N-1$ **do**
11: $\quad \ddot{h}^{(i)}(\mathbf{x}) = f_H^{(i)}(\mathbf{x}) \bar{*} h^{(i)}(\mathbf{x})$
12: **end for**
13: $\ddot{h}^{(N)}(\mathbf{x}) = f^{(N)}(\mathbf{x}) \bar{*} h^{(N)}(\mathbf{x})$
14: {Upsample and sum up the accelerations}
15: **for each** level $i = 2$ **to** N **do**
16: $\quad \ddot{H}^{(i)}(\mathbf{k}) = \mathcal{F}(\ddot{h}^{(i)}(\mathbf{x}))$
17: **end for**
18: **for each** level $i = N-1$ **to** 2 **do**
19: $\quad \ddot{H}^{(i)}(\mathbf{k}) \mathrel{+}= \text{upsample}(W_L^{(i+1)}(\mathbf{k})\, \ddot{H}^{(i+1)}(\mathbf{k}))$
20: **end for**
21: $\ddot{h}^{(1)}(\mathbf{x}) \mathrel{+}= \mathcal{F}^{-1}(\text{upsample}(W_L^{(2)}(\mathbf{k})\, \ddot{H}^{(2)}(\mathbf{k})))$
22: {Add other acceleration terms (obstacles, damping)}
23: {Perform numerical integration}

4.3 Full Pyramid Implementation

Given a precomputed pyramid dispersion kernel $\{f_H^{(1)}(\mathbf{x}) \dots f_H^{(N-1)}(\mathbf{x}), f^{(N)}(\mathbf{x})\}$, at runtime we compute wave accelerations by applying this pyramid kernel to the wave height field. However, instead of computing regular convolutions with spatially invariant kernels, we handle reflecting boundary conditions with arbitrary obstacles through the application of spatially variant kernels. We refer to this operation as *shadowed convolution* $f(\mathbf{x}) \bar{*} h(\mathbf{x})$, as it incorporates obstacle shadow information into the application of the dispersion kernels. Full details are provided in Section 5.

Before every downsampling or upsampling operation, we need to apply the low-pass filter $W_L(\mathbf{k})$ as discussed earlier. We have observed that spatial low-pass filtering with a compact filter produces aliasing artifacts that are perceptible in the simulation. Instead, we apply the full frequency-domain filter $W_L(\mathbf{k})$, which requires Fourier conversion from spatial to frequency domain and vice versa on each level of the pyramid. Once in the frequency domain, the downsample and upsample operations are trivial.

Figure 7: *Schematic depiction of three strategies to evaluate the modified height field $\bar{h}$ within the shadowed convolution (17), in an example where a blue obstacle intersects with the kernel window. The red pixel is the center of the kernel, where the acceleration is computed. The white pixel is one particular pixel that contributes to the convolution. From left to right: (i) reflection of height values about the obstacle normal (accurate but costly); (ii) zeroing of height values within the obstacle (cheap but inaccurate); (iii) zeroing of height values within obstacle shadows (our proposed compromise).*

With all these observations in mind, the full algorithm for simulation of dispersion waves follows the pseudocode in Algorithm 1. All inverse Fourier transforms in Step 7 can be executed in parallel, and similarly for all shadowed convolutions in Steps 11 and 13, and all Fourier transforms in Step 16. We take advantage of GPU streams to parallelize these operations in our implementation.

5 Interaction with Obstacles

The dispersion kernel assumes the existence of liquid in its whole domain. Obviously, the presence of obstacles invalidates this assumption, and the computation of accelerations turns into a spatially variant filtering operation. We focus on reflecting boundary conditions, and evaluate various approaches for spatially modulating the acceleration computation. We have found that modulation based on obstacle shadows yields an excellent balance between accuracy and computational cost. We also propose a massively parallelizable algorithm for the approximation of obstacle shadows, which minimizes the overhead w.r.t. the regular convolution operation.

5.1 Reflecting Boundary Conditions

Let us formally express the spatially variant modulation of acceleration computations in the presence of obstacles. As anticipated earlier, we refer to this operation as *shadowed convolution* $f(\mathbf{x}) \bar{*} h(\mathbf{x})$. We denote with $o \in \{0, 1\}$ a binary obstacle field, and with a $\bar{h}$ a modified height value affected by the obstacle field. Then, we express the spatially variant computation of acceleration using the shadowed convolution as:

$$\ddot{h}(\mathbf{x}) = f(\mathbf{x}) \bar{*} h(\mathbf{x}) = \sum_{\mathbf{z} \in \frac{D}{M}\,[-R \dots R]^2} f(\mathbf{z})\, \bar{h}(\mathbf{x}, \mathbf{z}, h, o). \quad (17)$$

In the absence of obstacles, $\bar{h}(\mathbf{x}, \mathbf{z}, h, o = 0) = h(\mathbf{x} - \mathbf{z})$, and the shadowed convolution becomes a regular convolution operation.

For planar obstacles larger than the dispersion kernel, reflecting boundary conditions can be trivially implemented by defining the modified height field $\bar{h}$ using height values replicated across obstacle boundaries [Bridson and Müller-Fischer 2007]. This approach could be extended to arbitrary obstacles and kernel sizes, by tracing a ray from $\mathbf{x}$ to $\mathbf{x} - \mathbf{z}$, and reflecting the ray about the obstacle

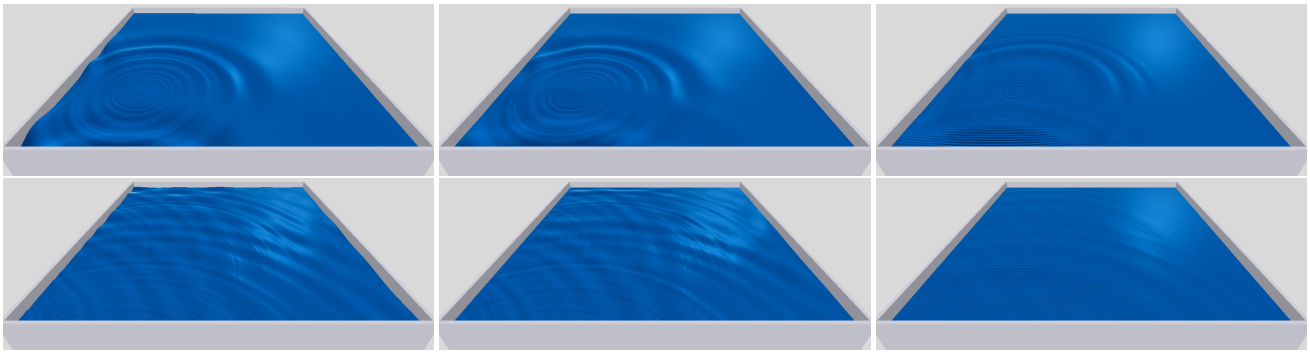


Figure 8: *The first two columns compare reflecting boundary conditions (left) vs. our shadow mask approach (middle). The behavior at the walls is slightly different, which can be seen on the top row,* 3.2 *seconds into the simulation. However, the approaches produce very similar wave frequency content, which is evident on the bottom row,* 8 *seconds into the simulation. The last column shows reflecting boundary conditions with the* iWave *method,* 20 *and* 30 *seconds respectively into the simulation, for comparison.*

normal at the first obstacle intersection. Unfortunately, this ray tracing approach adds a tremendous computational overhead on top of a regular convolution operation, in particular by adding branches during stream processing.

We propose an alternative for approximating reflecting boundary conditions, which defines the modified height $\bar{h}$ simply by masking the height value at locations that are under some obstacle's shadow. Formally, the modified height field is evaluated as:

$$\bar{h}(\mathbf{x}, \mathbf{z}, h, o) = \begin{cases} 0, & \exists \alpha \in [0,1] \text{ s.t. } o(\mathbf{x} - \alpha\,\mathbf{z}) = 1, \\ h(\mathbf{x} - \mathbf{z}), & \text{otherwise.} \end{cases} \tag{18}$$

Fig. 7 compares schematically the evaluation of the modified height for accurate reflecting boundary conditions, a simple obstacle mask that ignores the height value at locations occupied by obstacles, and our shadow mask criterion. Fig. 9 compares simulation results with the three approaches. With the obstacle mask approach, waves leak through thin objects. Note that the notion of 'thin' is relative to the wave numbers computed at each level of the pyramid, hence large objects may appear thin at low wave numbers. With our shadow mask approach, some waves appear slightly damped w.r.t. the accurate approach, but the overall behavior is well preserved.

Reflecting boundary conditions are a type of Neumann condition which constrain the energy flux through the boundary to be zero. Our shadow mask approach is in practice a Dirichlet condition, which forces reflected waves to have a negative amplitude such that the sum of incident and reflected waves cancel out. Up close, the two conditions may appear visually different, but at a distance they exhibit a very similar frequency spectrum and appear visually very similar, as shown in Fig. 8. The figure also shows reflecting boundary conditions with the *iWave* method, for comparison.

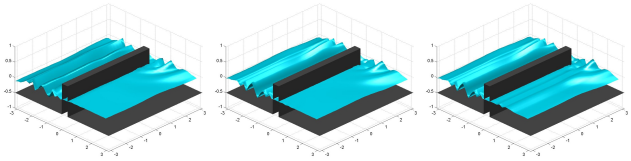


Figure 9: *Simulations of water waves passing through a slit on the far side, executed using different strategies for the approximation of reflecting boundary conditions. From left to right: (i) reflection of height values; (ii) our approach, zeroing of height values within obstacle shadows; (iii) zeroing of height values within obstacles. Our shadow approach approximates well the effects with accurate reflecting boundary conditions. By zeroing height values within obstacles, on the other hand, waves leak through the obstacles.*

5.2 Shadow Mask Propagation

Convolution operations are particularly well suited for stream processing. Multiple domain locations execute the convolution in parallel with minimal register pressure, efficient shared memory usage, and no branching. We aim at designing a shadowed convolution operation that minimizes the computational overhead w.r.t. the regular convolution operation.

Given a grid point $\mathbf{x}$ where the acceleration is computed, the rationale of our efficient shadowed convolution operation is to traverse the convolution window outward from $\mathbf{x}$, propagating the shadow mask in conjunction with the evaluation of the kernel multiplications. Specifically, we partition the convolution window into four equal triangles (top, bottom, left, and right). As shown in Fig. 10, for the top triangle we propagate the shadow mask upward one row at a time. We approximate the true shadow propagation by defining four radial sectors within the triangle, and for each sector two dif-

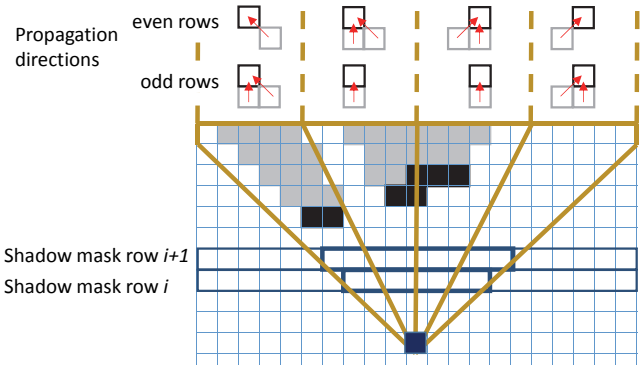


Figure 10: *Shadows are propagated row by row, computing the shadow mask for row $i+1$ as a Boolean function of the obstacle mask in row $i+1$ and the shadow mask in row i. The top triangle of the kernel window (and similarly for the rest) is divided into four sectors. Two different shadow propagation rules, for even and odd rows, are defined on each sector, to approximate accurate shadows with a minimal computational cost.*

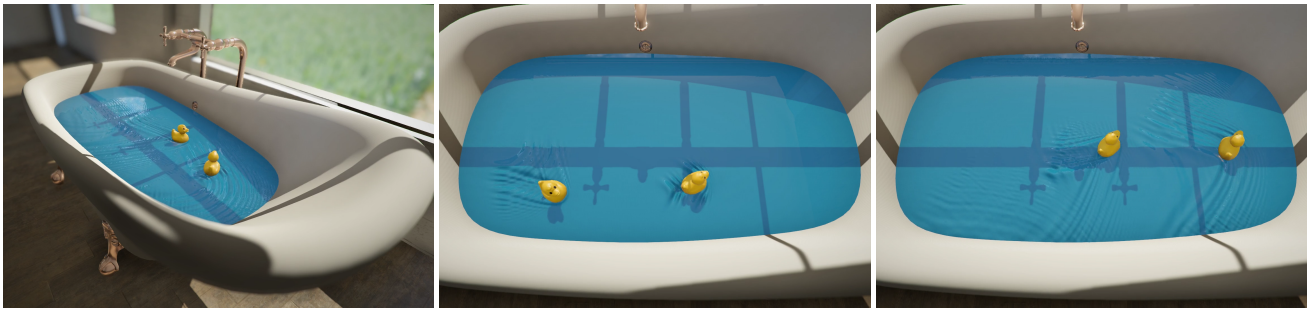


Figure 11: *Rubber ducks swim in a bathtub and generate wakes that exhibit dispersion into slow gravity waves and faster capillary waves. The domain is* 0.75 *meters long, and is simulated on a* 512×512 *grid, at* 400 *ms per frame.*

ferent propagation directions for odd and even rows. Fig. 10 shows two example shadows approximated with our efficient propagation approach. We store each row's shadow mask as a bit-mask, and the shadow mask for row $i+1$ is computed as a Boolean function of the obstacle field for row $i+1$ and the shadow mask for row i. The particular Boolean function for each sector and row depends on the active propagation directions, and is efficiently implemented using additional masks.

Our efficient shadow propagation closely approximates accurate shadows, and hence our shadowed convolution operation succeeds at modeling reflecting boundary conditions. This is achieved with minimal overhead w.r.t. regular convolution. Specifically, the existing overhead is due to the modified traversal of the kernel window, the propagation of the shadow mask (one Boolean function per row/column), and the modulation of the kernel multiplications with shadow bits.

6 Results

We have demonstrated the application of our wave simulation method on several animation examples. In all of them, simulations were executed on a 4.0 GHz Quad-core Intel Core i7-4790K CPU with 16GB of memory, and a NVIDIA GTX970 graphics card with 4GB of memory. For all the examples, the kernel pyramid has been constructed using 32×32 kernels on each level. We have used symplectic Euler integration, with a tiny amount of damping. Under CFL conditions, the simulations are stable; we add damping only to slowly relax the waves over time.

Statistics and Performance In Table 1, we summarize the simulation settings and performance in our major experiments. The average simulation cost is of roughly 200 ms per time step for a 1024×1024 grid. The time step requirements depend on the size and resolution of the domain, as well as the water depth. Together, they determine the range of wave lengths represented in the simulation, and thus the range of propagation velocities. In our examples, the time step varies between 2 ms for scenes with a resolution of 1 mm, and 16 ms for a scene with a resolution of 15 mm. Performance varies then between 0.6 fps and 5 fps (for $1/60$ sec. frames). The dominant cost of our simulations is the shadowed convolution operations, which take 64% of the time on 1024×1024 grids, followed by FFT and IFFT operations, which take 28% of the time.

Scene	Size (m)	Depth (m)	Grid size	Time step (ms)	Cost/step (ms)
Pond	4	6	1024	2	201
Bathtub	0.75	0.2	512	2	53
Canal	15	10	1024	16	192

Table 1: *Scene settings and performance statistics for our major experiments.*

Scale-Dependent Effects The pond scene in Fig. 1 and the bathtub scene in Fig. 11 show a combination of short gravity waves and capillary waves. When the paper birds and the ducks produce wakes, thin capillary waves travel faster outward. The pond scene also shows circular waves produced by rain drops, again with characteristic capillary effects. All waves, short and long, are reflected efficiently on both static and dynamic objects, thanks to the efficient handling of reflecting boundary conditions. The canal scene in Fig. 13 shows longer gravity waves, which also reflect on the objects in the scene.

Shallow Water and Refraction Our method can easily be adapted to incorporate non-uniform water depth, following the depth-based kernel interpolation approach of Tessendorf [2004b]. We have validated this extension, which incurs only a performance penalty of 22%. The extension succeeds at capturing the characteristic refraction and steepening effects of the shallow water regime, as shown in Fig. 12.

Analysis of Wave Transfer As mentioned in Section 4.2, the shadowed convolution operation breaks the linearity and spatial invariance assumptions made in our pyramid kernel design. As a result, the low-pass filtering of height fields carried out in the construction of the height pyramid is not fully compensated during the reconstruction of accelerations. The height field is effectively blurred in space prior to convolution with the pyramid kernel, and this blur may transfer waves through thin obstacles. Similarly, on the reconstruction step, the same blur effect may transfer wave ac-

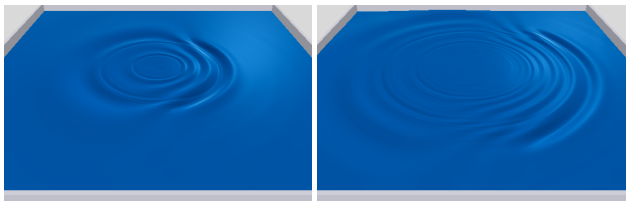


Figure 12: *Our method captures the characteristic refraction and steepening effects of shallow water. In the example, water height ranges from* 10 *m on the left side to just* 10 *cm on the right side of a* 2*-meter-wide domain. The snapshots are taken* 1.3 *and* 2.2 *seconds into the simulation respectively.*

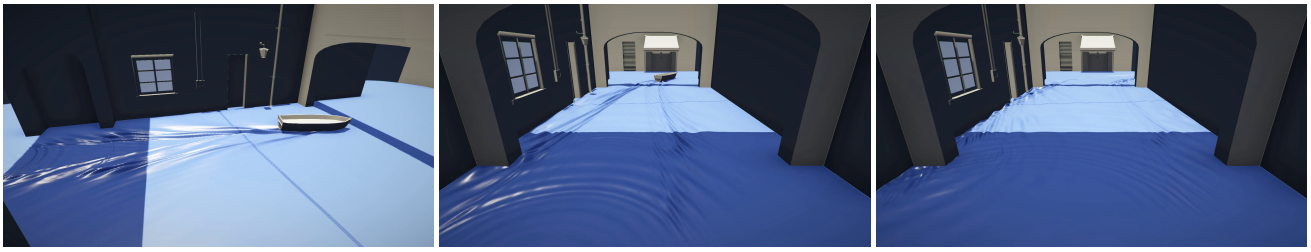


Figure 13: *A boat plows through a canal and generates wakes that reflect on the walls. The domain is* 15 *meters long, and is simulated on a* 1024×1024 *grid, at just* 200 *ms per frame.*

celerations through thin obstacles.

We have evaluated this wave transfer effect, and we have found that it is negligible in practical settings, far milder than the leaking suffered by the obstacle-mask approximation of reflecting boundary conditions shown in Fig. 9. We have designed a worst-case scenario to analyze the magnitude of wave transfer, with two pools separated by a perfect wall, just 5 cells wide, on a single 512×512 simulation grid. We start a wave on one of the pools and evaluate the amount of energy transferred to the other pool over time. After one minute of simulation time, less than 5% of the energy has transferred to the wrong side, which is barely visible, as shown in Fig. 14. We conclude that, in combination with all the wave dynamics in a regular animation, wave transfer is imperceptible.

7 Discussion and Future Work

We have presented the first method to accurately simulate dispersing height field waves with obstacle interactions in the spatial domain. This is made possible with a carefully constructed, error-compensated dispersion kernel, and a fast, GPU-friendly obstacle shadowing technique. In conjunction, these contributions enable efficient simulations of a wide range of wave phenomena with complex obstacles.

Our method makes certain approximations that could affect the realism of the animated waves. First and foremost, our method simulates the Airy linear wave model, hence nonlinear effects are simply not accounted for. Reflecting boundary conditions are also approximated, although we have demonstrated that our shadow mask approximation produces qualitatively similar results. In addition, as discussed earlier, our downsampling and upsampling operations lead to some wave transfer across thin obstacles. Instead of our current frequency-domain low-pass filtering, we plan to investigate customized spatial filters that prevent wave transfer altogether.

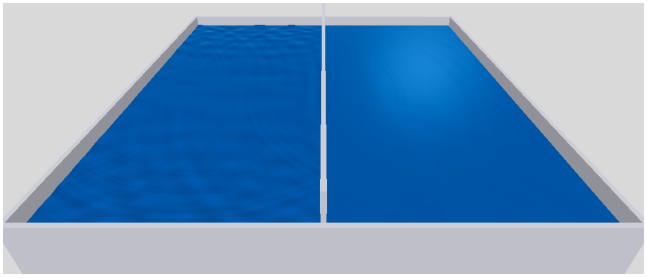


Figure 14: *Evaluation of wave transfer on worst-case scenario. A wave is induced on the left pool, and after one minute of simulation time less than* 5% *of the energy is transferred to the right pool through the thin wall.*

Other parts of our algorithm also make assumptions that leave room for improvement. E.g., our damping currently does not consider wave lengths, and dampens the whole spectrum uniformly. For obstacles, we currently assume that they extend all the way to the ground. All of these areas will be interesting avenues for refinement in the future,

We are also interested in adding a sharpening filter to our simulations, as it is commonly done for spectral wave synthesis [Tessendorf 2004b]. This will make it possible to realize more generic, trochoidal wave shapes.

In the future, we are excited about the outlook to tightly couple accurate, height field wave simulations with three-dimensional solvers. Our approach could provide the missing link to smoothly transition from ocean waves synthesized with spectral approaches, to regular 3D liquid simulations.

Acknowledgements

We would like to thank Daniel Lobo, Rosa Sánchez and Héctor Barreiro for help with demo production, and Chris Wojtan and the anonymous reviewers for useful feedback. This work was supported in part by the Spanish Ministry of Economy (grants TIN2014-62143-EXP and TIN2015-70799-R), the European Research Council (ERC-2011-StG-280135 Animetrics and ERC-2015-StG-637014 realFlow), and a National Science Foundation CAREER award (IIS-1253948). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

AIRY, G. 1849. *Tides and Waves*. J.J. Griffin.

BRIDSON, R., AND MÜLLER-FISCHER, M. 2007. Fluid simulation. In *ACM SIGGRAPH 2007 Courses*.

BRIDSON, R. 2015. *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis.

BURT, P., AND ADELSON, E. 1983. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications 31*, 4, 532–540.

CHENTANEZ, N., AND MÜLLER, M. 2010. Real-time simulation of large bodies of water with small scale details. In *Symposium on Computer Animation*, 197–206.

CORDS, H. 2008. Moving with the flow: Wave particles in flowing liquids. In *Winter School of Computer Graphics (WSCG)*.

DARLES, E., CRESPIN, B., GHAZANFARPOUR, D., AND GONZATO, J. 2011. A Survey of Ocean Simulation and Rendering

Techniques in Computer Graphics. *Comput. Graph. Forum 30*, 43–60.

DEAN, R., AND DALRYMPLE, R. 1991. *Water Wave Mechanics for Engineers and Scientists*. World Scientific.

ENRIGHT, D., NGUYEN, D., GIBOU, F., AND FEDKIW, R. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *ASME/JSME 2003 4th Joint Fluids Summer Engineering Conference*, American Society of Mechanical Engineers, 337–342.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proc. of SIGGRAPH*, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models Image Process. 58* (September).

IHMSEN, M., ORTHMANN, J., SOLENTHALER, B., KOLB, A., AND TESCHNER, M. 2014. SPH fluids in computer graphics. In *Eurographics - State of the Art Reports*, 21–42.

JESCHKE, S., AND WOJTAN, C. 2015. Water wave animation via wavefront parameter interpolation. *ACM Transactions on Graphics 34*, 3, 27.

KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. In *Proc. of SIGGRAPH*.

KIM, T., TESSENDORF, J., AND THUEREY, N. 2013. Closest point turbulence for liquid surfaces. *ACM Transactions on Graphics 32*, 2.

LAMB, H. 1994. *Hydrodynamics*, 6 ed. Cambridge University Press.

LOVISCACH, J. 2002. A convolution-based algorithm for animated water waves. In *Eurographics Short Presentations*, 381–389.

MERCIER, O., BEAUCHEMIN, C., THUEREY, N., KIM, T., AND NOWROUZEZAHRAI, D. 2015. Surface turbulence for particle-based liquid simulations. *ACM Transactions on Graphics 34*, 6, 202.

NIELSEN, M. B., AND BRIDSON, R. 2011. Guide shapes for high resolution naturalistic liquid simulation. In *ACM Transactions on Graphics (TOG)*, vol. 30, ACM, 83.

NIELSEN, M. B., SÖDERSTRÖM, A., AND BRIDSON, R. 2013. Synthesizing waves from animated height fields. *ACM Transactions on Graphics (TOG) 32*, 1, 2.

OTTOSSON, B. 2011. *Real-time Interactive Water Waves*. Master's thesis, KTH.

PATEL, S., TESSENDORF, J., AND MOLEMAKER, J. 2009. Monocoupled 3D and 2D river simulations. In *Symposium on Computer Animation, Poster*.

SIMONCELLI, E. P., AND FREEMAN, W. T. 1995. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *Image Processing, 1995. Proceedings., International Conference on*, vol. 3, 444–447.

SOLENTHALER, B., BUCHER, P., CHENTANEZ, N., MÜLLER, M., AND GROSS, M. 2011. Sph based shallow water simulation. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)*.

STAM, J. 1999. Stable fluids. In *Proc. of SIGGRAPH*, 121–128.

TESSENDORF, J. 2004. *Interactive Water Surfaces*. Charles River Media.

TESSENDORF, J. 2004. Simulating ocean surfaces. *SIGGRAPH Course*.

TESSENDORF, J., 2008. Vertical derivative math for iwave.

THACKER, J. 2010. Go with the flow. *3D World* (Sept.).

THUEREY, N., AND PFAFF, T., 2016. MantaFlow. *http://mantaflow.com*.

THÜREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A multiscale approach to mesh-based surface tension flows. *ACM Transactions on Graphics 29* (July), 48:1–48:10.

WANG, H., MILLER, G., AND TURK, G. 2007. Solving general shallow wave equations on surfaces. In *Symp. on Computer Animation*.

YU, J., WOJTAN, C., TURK, G., AND YAP, C. 2012. Explicit mesh surfaces for particle based fluids. *Comp. Graph. Forum*.

YUKSEL, C., HOUSE, D. H., AND KEYSER, J. 2007. Wave particles. *ACM Transactions on Graphics 26*, 3 (July).