

Wave Curves: Simulating Lagrangian water waves on dynamically deforming surfaces

TOMAS SKRIVAN, IST Austria, Austria
ANDREAS SODERSTROM, no affiliation, Sweden
JOHN JOHANSSON, Weta Digital, New Zealand
CHRISTOPH SPRENGER, Weta Digital, New Zealand
KEN MUSETH, Weta Digital, New Zealand
CHRIS WOJTAN, IST Austria, Austria

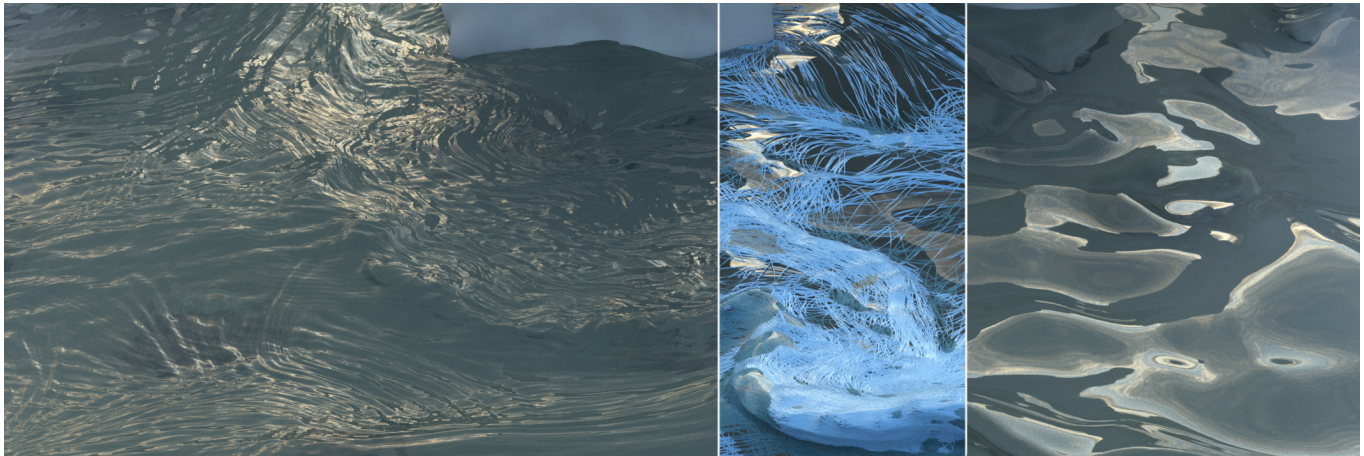


Fig. 1. We introduce an efficient method for adding detailed ripples (left) in the form of curve primitives (middle right) on top of an existing 3D fluid simulation (right). These wave curves evolve according to our extension of linear water wave theory, which naturally models effects like ripples aligned with flow features.

We propose a method to enhance the visual detail of a water surface simulation. Our method works as a post-processing step which takes a simulation as input and increases its apparent resolution by simulating many detailed Lagrangian water waves on top of it. We extend linear water wave theory to work in non-planar domains which deform over time, and we discretize the theory using Lagrangian wave packets attached to spline curves. The method is numerically stable and trivially parallelizable, and it produces high frequency ripples with dispersive wave-like behaviors customized to the underlying fluid simulation.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; *Simulation by animation*.

Authors' addresses: Tomas Skrivan, IST Austria, Am Campus 1, Klosterneuburg, 3400, Austria, tomas.skrivan@ist.ac.at; Andreas Soderstrom, no affiliation, Sweden, andreas.soderstrom@gmail.com; John Johansson, Weta Digital, New Zealand, jjohansson@wetafx.co.nz; Christoph Sprenger, Weta Digital, New Zealand, christophspr@gmail.com; Ken Museth, Weta Digital, New Zealand, ken.museth@gmail.com; Chris Wojtan, IST Austria, Am Campus 1, Klosterneuburg, 3400, Austria, wojtan@ist.ac.at.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2019 Copyright held by the owner/author(s).

0730-0301/2019/11-ART65

<https://doi.org/10.1145/3386569.3392466>

Additional Key Words and Phrases: Water surface waves, wave animation, production animation, ripples, wakes, dispersion

ACM Reference Format:

Tomas Skrivan, Andreas Soderstrom, John Johansson, Christoph Sprenger, Ken Museth, and Chris Wojtan. 2019. Wave Curves: Simulating Lagrangian water waves on dynamically deforming surfaces. *ACM Trans. Graph.* 38, 6, Article 65 (November 2019), 12 pages. <https://doi.org/10.1145/3386569.3392466>

1 INTRODUCTION

Although three-dimensional fluid simulation has led to spectacularly detailed visual effects in the past decade, the expense of three-dimensional fluid simulation strongly limits the amount of detail that can be simulated at the water surface. Several researchers have circumvented this limitation by simulating additional 2D waves directly on top of the 3D water surface. However, these previous approaches have a few significant limitations, like a resolution limit imposed by an underlying grid, mesh, or particle system [Angst et al. 2008; Kim et al. 2013; Mercier et al. 2015; Thürey et al. 2010; Yang et al. 2016; Yu et al. 2012], or a restriction to perfectly flat domains [Canabal et al. 2016; Jeschke et al. 2018; Jeschke and Wojtan 2015, 2017; Yuksel et al. 2007].

In this work, we aim to significantly increase the visible detail on a simulated fluid surface using Lagrangian wave packets, which

de-couple the wave resolution from the simulation resolution. We first derive evolution equations for water waves on a moving surface (e.g. an existing fluid simulation or a hand-animated surface), taking into account the physical effect of the surface's acceleration on both the wavelength and wave speeds. We then discretize the continuous theory with a new Lagrangian *wave curve* simulation primitive, enabling stable, resolution-independent, parallel simulations of coherent connected wave structures on a flowing surface (See Figure 1). These wave curves provide us with a mechanism to greatly enhance the visual detail of a fluid surface in a physically plausible manner, with little computational expense.

We summarize the contributions of our method as follows:

- The first technique based on Lagrangian wave packets for adding wave detail to an existing fluid simulation.
- An extension of Airy's theory for water waves on dynamically deforming water surfaces.
- The introduction of curve-shaped wave packets.

2 RELATED WORK

A surge of research on water animation for visual effects has created a substantial body of publications, and we shall only discuss the ones most related to ours — lower dimensional techniques for simulating water surfaces, and methods for enhancing full three dimensional free-surface water simulations.

Methods for solving surface water waves as height-fields defined on horizontal 2D domains typically lead to superior computational performances and domain sizes, but they are naturally limited by the underlying lower dimension representation, which cannot capture overturning and splashing. Examples include the early work by Kass and Miller [1990], which allowed for simple ripple effects to be simulated by solving the scalar shallow wave equation, the simulation of ocean waves by means of 2D spectral methods [Tessendorf 2002, 2004], Lagrangian wave particles [Yuksel et al. 2007], wavefront tracking [Jeschke and Wojtan 2015], wave kernels that better capture dispersive effects [Canabal et al. 2016; Loviscach 2002; Ottosson 2011], structure preserving integrators for the Euler-Poincaré differential equation [Azencot et al. 2018], and water surface wavelets [Jeschke et al. 2018]. Wang et al. [2007] proposed an improved height-field approach, where the general shallow wave equation is solved in the normal direction of non-planar surfaces, alleviating some of the limitations of Kass and Miller [1990]. The wave solver most closely related to ours is that of water wave packets [Jeschke and Wojtan 2017]. Their work efficiently animates high frequency wave details with physically consistent wave speeds. However, the method assumes a static horizontal planar water surface, and they offer no indication how to extend the method to work on the surface of a moving 3D fluid simulation.

Another common practice in VFX is to employ a hybrid approach where 2D and 3D water simulations, applied to distinct domains, are seamlessly coupled. For instance Irving et al. [2006] and Thürey et al. [2006] couple a 2D deep water solver using tall cells with a full 3D free-surface solver for the actual water-air interface, whereas Cords [2008] couples the wave particles of Yuksel et al. [2007] with a 3D free-surface solver.

Prior to our work, many previous researchers have designed algorithms for simulating water waves on top of an existing free-surface 3D water simulation. Such an approach introduces increased visual details without the computational cost that comes from attempting to capture these details directly in the 3D solver. One of the first publications to explore this idea was Patel et al. [2009], who mono-coupled a 2D iWave simulation onto a 3D liquid simulation with a 2D orthographic projection mapping. Subsequently, Thürey et al. [2010] simulated the constant-speed wave equation directly on a Lagrangian fluid surface mesh, focusing on surface tension effects. They also coupled the added higher frequency detail back to the coarse 3D simulation. Yu et al. [2012] used the thin plate equation for surface waves on a moving mesh, and Angst et al. [2008] solved the shallow wave equation on an animated mesh model (as opposed to a free-surface from a 3D water simulation). Bojsen-Hansen et al. [2012] simulated waves on arbitrary deforming meshes with changing topology, using fluid simulation as an example. Bojsen-Hansen and Wojtan [2013] simulated non-linear waves inspired by vortex sheets to correct defects in a low-resolution 3D water surface. Kim et al. [2013] solved water waves with a more realistic dispersion relation on top of a free-surface, avoiding the problem of surface parameterization by simulating waves directly on the deforming level set surfaces. Mercier et al. [2015] bypassed the generation of level set surfaces and adds turbulent details directly to the particles resulting from a 3D FLIP simulation. Yang et al. [2016] adds the high-frequency details of capillary waves to 3D SPH simulations by converting surface tension energy changes to density variations.

The methods described above model waves with a diverse range of physical behaviors. Most approaches use a simple constant-speed wave equation or a thin plate equation, which are arguably more appropriate for elastic wave simulation. In contrast, the work of Kim et al. [2013] models waves with a dispersion relation derived for water surface dynamics. Our work builds upon this idea by adding an additional term to the dispersion relation to model the influence of surface accelerations. Bojsen-Hansen and Wojtan [2013]'s non-linear vortex sheet approach is also based on water surface dynamics, but its non-linearity requires smaller time steps. Most importantly from our perspective, all of the existing methods for animating water waves on a moving surface use an Eulerian discretization, so the visible wave detail will be limited by the density of simulation degrees of freedom (grid nodes or particles). In contrast, our wave curve primitives add high-frequency visual detail independent of a simulation resolution.

A full discussion of the physics underlying water surface waves is outside of the scope of this paper. The theory of linearized water waves was first described by Airy [1841]. The book by Whitham [1999] provides excellent and understandable descriptions of both linear and non-linear waves (including water waves), and the book by Mei [2005] works out details for waves with a background flow and variable depth.

3 THEORY

To describe the dynamics of water waves, we use linearized wave theory [Airy 1841] which accurately models many physical properties of small amplitude waves such as frequency, speed, and dispersion.

However, this standard theory is concerned only with waves on a horizontal water surface, while we are interested in small waves on moving surfaces where many additional effects are present. Moving surfaces can change the wave dynamics in non-trivial and visually fascinating ways: simply angling the surface so that its normal does not align with gravity changes the speed of gravity waves, but has no effect on capillary waves; curvature of a surface can focus or de-focus waves, because waves follow geodesic paths; accelerating the surface in the normal direction can speed up or slow down waves; and movement of the surface in the tangent direction can stretch, compress, shear, and bend waves.

In Section 3.1, we first give an overview of the basic linear wave theory on a flat water surface where any wave can be written as a linear combination of planar waves. Next, in Section 3.2, we expand the discussion to include waves with a background flow. Building on these ideas, we derive a linear wave theory on moving surfaces in Section 3.3.

3.1 Linear wave theory

Here we outline the basic notions behind Airy wave theory. For a more accessible and complete introduction to the linear wave theory, see, for example, [Acheson 1990, Chapter 3].

Airy wave theory defines the water height η of a single wave as

$$\eta(\mathbf{x}, t) = A \cos(\mathbf{k} \cdot \mathbf{x} - \omega t + \theta_0), \quad (1)$$

where A is the wave amplitude, $\mathbf{x} \in \mathbb{R}^2$ is a point in a plane, $\mathbf{k} \in \mathbb{R}^2$ is the wavevector defining the wave's traveling direction, $k = \|\mathbf{k}\|$ is the wavenumber, $\lambda = \frac{2\pi}{k}$ is the wavelength, ω is the frequency, and θ_0 is the phase shift. The main result of the linear wave theory is the dispersion relation connecting the wavenumber k with the frequency ω .

$$\omega = \Omega(\mathbf{k}) = \sqrt{\left(g + \frac{\gamma}{\rho} k^2\right)} k, \quad (2)$$

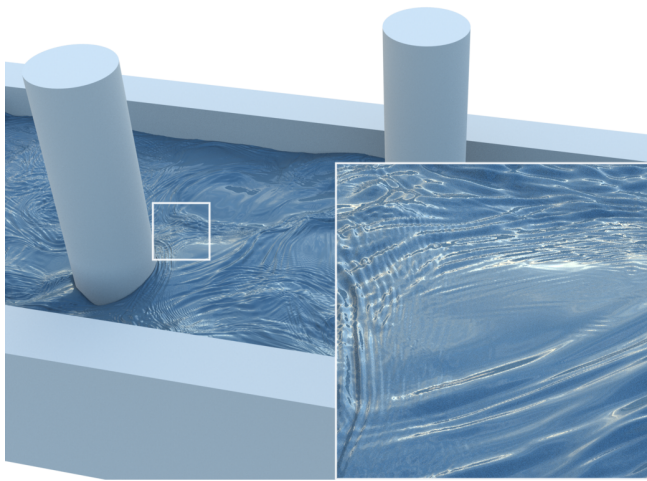


Fig. 2. A close-up on detail of wave simulation in the canal scene.

where g is the gravitational acceleration, γ is the surface tension and ρ is the water density. Throughout this exposition, we will often use the shorthand Ω for readability, instead of writing out $\Omega(\mathbf{k})$. Similarly, although $\Omega(\mathbf{k})$ takes a wavevector as its argument, it only computes with its magnitude $k = \|\mathbf{k}\|$, so we will sometimes write $\Omega(k)$ in equations that otherwise deal entirely with scalars.

The energy density of a wave is

$$E = \frac{1}{2} \left(\rho g + \gamma k^2 \right) A^2 = \frac{\rho \Omega^2}{2k} A^2 \quad (3)$$

which is transported with the group speed

$$\mathbf{c}_g = \frac{\partial \Omega}{\partial \mathbf{k}} = \frac{\partial \Omega}{\partial k} \hat{\mathbf{k}}. \quad (4)$$

This transportation of energy density indirectly describes how wave amplitudes propagate over the surface. It will be used for computing both amplitude propagation and wave seeding later in this paper.

3.2 Linear wave theory in slowly varying environment

In this section we give a summary of results from [Mei 2005, Section 3.6] where they examine the effect of a current and varying water depth on water wave dynamics. We omit the effects of water depth in this discussion, because we wish to study waves whose wavelength is small compared to the typical length scales of a fluid simulation.

In this case, we can no longer represent η with plane waves as in (1), because the wavefronts can be stretched or bent by a background current. Instead, the waves are represented in a more general form

$$\eta(\mathbf{x}, t) = A(\mathbf{x}, t) \cos \theta(\mathbf{x}, t) \quad (5)$$

where θ is the phase function. This general form of a wave looks like a planar wave only locally—We can see this by Taylor expanding the phase function:

$$\theta(\mathbf{x}, t) \approx \theta(\mathbf{x}_0, t_0) + \frac{\partial \theta}{\partial \mathbf{x}}(\mathbf{x}_0, t_0) \cdot (\mathbf{x} - \mathbf{x}_0) + \frac{\partial \theta}{\partial t}(\mathbf{x}_0, t_0) (t - t_0). \quad (6)$$

Around the point (\mathbf{x}_0, t_0) the wave (5) looks like the planar wave in (1) with wavevector $\mathbf{k}(\mathbf{x}_0, t_0)$ and frequency $\omega(\mathbf{x}_0, t_0)$ defined as

$$\mathbf{k}(\mathbf{x}, t) = \frac{\partial \theta}{\partial \mathbf{x}}(\mathbf{x}, t) \quad \omega(\mathbf{x}, t) = -\frac{\partial \theta}{\partial t}(\mathbf{x}, t) \quad (7)$$

Here, ω is the *absolute frequency* measured by a stationary observer—an observer moving along a current will measure a different frequency, referred to as the *intrinsic frequency*, σ . The relation between the two frequencies is

$$\sigma = -\frac{D\theta}{Dt} = \omega - \mathbf{k} \cdot \mathbf{U} \quad (8)$$

where \mathbf{U} is the current velocity and $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{U} \cdot \nabla$ is the convective derivative. The intrinsic frequency is the physical frequency of the wave as it moves along the surface, so it is important that we use σ instead of ω in the dispersion relation (2) for waves on a moving surface. Thus, in the presence of a current, the dispersion relation is:

$$\sigma = \Omega(k) = \sqrt{\left(g + \frac{\gamma}{\rho} k^2\right)} k. \quad (9)$$

One interesting consequence of allowing an environment that varies over time is that the energy of an individual wave is no longer conserved — the total energy of the entire system (both waves and current combined) is conserved, but there can be an energy transfer between the waves and the environment. However, a related quantity, the *wave action*, is still conserved. The wave action \mathcal{A} is defined as

$$\mathcal{A} = \frac{E}{\Omega} = \frac{\rho\Omega}{2k} A^2, \quad (10)$$

and we can write out its conservation law as it is advected by the current velocity \mathbf{U} and the group speed \mathbf{c}_g :

$$\frac{\partial \mathcal{A}}{\partial t} + \text{div}((\mathbf{U} + \mathbf{c}_g) \mathcal{A}) = 0 \quad (11)$$

The conservation of the wave action is actually a very general principle and holds for many linear and non-linear physical systems exhibiting wave-like motion. For more details, see [Whitham 1999, Chapter 14].

3.3 Linear wave theory on moving surfaces

Now that we have established the existing background on linear water surface waves, we can extend these ideas to surfaces which have non-zero curvature and can deform over time.

As above, we describe a wave on a moving surface with

$$\eta(\mathbf{x}, t) = A(\mathbf{x}, t) \cos \theta(\mathbf{x}, t) \quad (12)$$

but we generalize the water height η to be a deflection of the surface in the normal direction (not necessarily a vertical displacement). Similarly, the point \mathbf{x} is now a point on the surface, and any spatial gradient has to be interpreted as a surface gradient. Therefore the wavevector \mathbf{k} is a tangent vector to the surface. For any point on the surface, we can create a local reference frame whose upward direction is aligned with the surface normal. However, if the surface is moving, then this is a *non-inertial* reference frame. In addition to the results from Section 3.2, it will experience a fictitious inertial force, requiring us to add an acceleration that is equal and opposite to the acceleration of the coordinate frame.

In effect, we only modify the theory from Section 3.2 by removing the assumption that gravity opposes the normal direction, and by replacing every occurrence of the gravity g with the *effective gravity*

$$g^*(\mathbf{x}, t) = -\mathbf{N}(\mathbf{x}, t) \cdot (\mathbf{g} - \mathbf{a}(\mathbf{x}, t)), \quad (13)$$

with surface normal \mathbf{N} and surface acceleration \mathbf{a} . The dispersion relation on moving surfaces becomes

$$\sigma = \Omega(k, g^*) = \sqrt{\left(g^* + \frac{\gamma}{\rho} k^2\right) k}. \quad (14)$$

The conservation of the wave action (11) remains unchanged, but we have to use the new dispersion relation in the definitions of action \mathcal{A} and group speed \mathbf{c}_g .

Replacing gravity g with the local gravity g^* in the dispersion relation (14) has some interesting new consequences that were not present in Airy's original theory. Wave speeds increase when the surface accelerates upward, and they slow down when the surface falls downward. Extremely fast downward accelerations can even cause the dispersion relation to become imaginary when $g^* + \frac{\gamma}{\rho} k^2 < 0$.

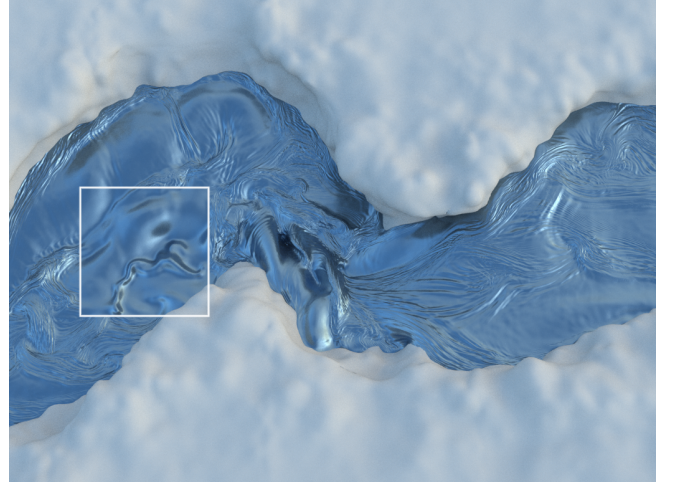


Fig. 3. Wave curves add rich details to a 3D river simulation. The detail from the underlying fluid simulation is seen in the white box.

This imaginary σ figuratively throws a wrench into the wave dynamics, causing the traveling wave to slow to a halt while the amplitude blows up exponentially. This phenomenon is known as the *Rayleigh-Taylor instability* and the standard stability analysis [Drazin 2002, Section 3.7] reveals that the condition for the instability is exactly $g^* + \frac{\gamma}{\rho} k^2 < 0$, consistent with our analysis here.

We note that other researchers have also investigated this concept of effective gravity outside of the computer animation discipline. Longuet-Higgins [1985] investigated the effective gravity of waves riding on top of long-wavelength gravity waves, and Longuet-Higgins [1995] investigated the analogous case for capillary waves riding on a Stokes wave. Their computations of the analytic acceleration of such surface motions yields η_{xx} or curvature-dependent terms in g^* , respectively. Our work does not make assumptions about the underlying surface motion, so we use a generic term in g^* instead.

We now have a theoretical foundation for linear water waves on moving surfaces, and it is consistent with a observable phenomena like known wave travel speeds and Rayleigh-Taylor instabilities. From these foundations, we derive equations of motion in Section 3.4, and we discuss conditions for wave amplification in Section 3.5.

3.4 Lagrangian view

In the end, we wish to track many *wave curves* on top of the moving surface. Each curve is just a set of Lagrangian particles which carry information about the waves (similar to the “Wave Packets” approach of Jeschke et al. [2017]). Using the method of characteristics, we can derive from the dispersion relation (14) that these Lagrangian particles move with velocity $\mathbf{U} + \mathbf{c}_g$ and carry the value of the wavevector $\mathbf{k}(t)$, frequency $\omega(t)$ and phase $\theta(t)$. The detailed calculation can be found in Appendix A. The particle position $\mathbf{x}(t)$ satisfies

$$\dot{\mathbf{x}} = \mathbf{U} + \mathbf{c}_g \quad (15)$$

and the other quantities evolve in the following way

$$\dot{\mathbf{k}} = -\frac{\partial \Omega}{\partial g^*} \nabla g^* - [\nabla \mathbf{U}]^T \mathbf{k}, \quad (16)$$

$$\dot{\theta} = -\Omega + \mathbf{c}_g \cdot \mathbf{k}. \quad (17)$$

These equations can be easily time-stepped, and we can reconstruct the phase function around each Lagrangian particle using a local Taylor expansion. We could also derive an evolution equation for the wave action, but we found it more convenient to use the conservation law (11), which says that the wave action is constant over a patch $\Sigma(t)$ which is advected with the velocity $\mathbf{U} + \mathbf{c}_g$, i.e.

$$\int_{\Sigma(t)} \mathcal{A} dx = \text{constant}. \quad (18)$$

The discrete treatment of these equations will be discussed in more detail in Section 4.

3.5 Wave growth

Once we know how these waves evolve, it is natural for us to ask how they are created in the first place. Unfortunately, linear wave theory cannot predict the formation of waves from nothing; it only describes how already-existing waves will grow or shrink over time depending on their environment. In the absence of a sound theory for predicting the creation of water waves, our idea is to instead identify which scenarios would cause an invisibly small wave ($kA \ll 1$) to quickly grow in energy.

We derive an equation for the evolution of surface wave energy by writing the conservation law for wave action in terms of energy (inserting (9) into (11)). For detailed calculation, see Appendix B. We obtain the following relation:

$$\frac{\partial E}{\partial t} + \text{div}((\mathbf{U} + \mathbf{c}_g) E) = \frac{E}{\sigma} \left(-\frac{c_g}{c_p} \hat{\mathbf{k}} \cdot \mathbb{D} \hat{\mathbf{k}} + \frac{1}{\sigma} \frac{\partial \Omega}{\partial g^*} \frac{Dg^*}{Dt} \right) \quad (19)$$

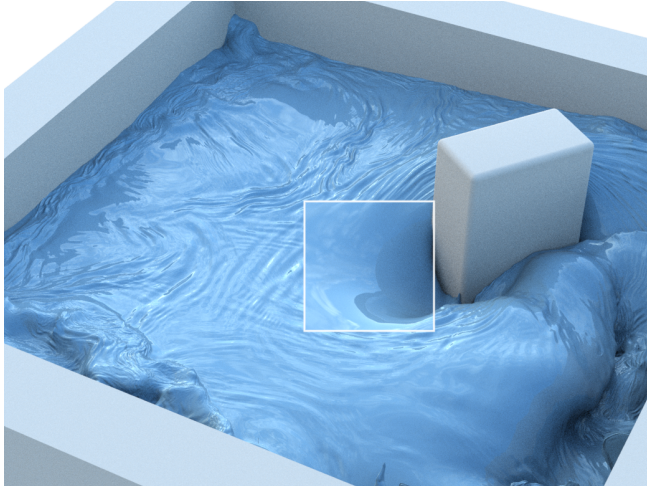


Fig. 4. Similar to previous methods, we add detail to a pool churned up by a rotating object. The white inset box shows the details from the underlying FLIP simulation.

where $c_p = \frac{\Omega}{k}$ is the phase speed, $\hat{\mathbf{k}} = \mathbf{k}/k$ is a unit vector in the direction of \mathbf{k} , $\mathbb{D} = \frac{1}{2} (\nabla \mathbf{U} + \nabla \mathbf{U}^T)$ is the symmetric part of the velocity gradient, and “div” is the divergence restricted to the fluid surface (\mathbf{U} for an incompressible velocity field can still have a non-zero surface divergence, for example in regions where the flow rises from below and spreads apart at the surface).

This equation states the energy balance for our water surface waves. The left hand side is in conservation law form, so non-zero terms on the right side represent external sources of wave energy transferred to the surface, triggered by changes in its environment (changes in the surface caused by the velocity field \mathbf{U} inside \mathbb{D} , and changes in the effective gravity g^* caused by the accelerating fluid surface). We first note that the energy E is proportional to its own time derivative, so we can interpret the multiplied term as an energy growth rate, G :

$$G(\mathbf{x}, \mathbf{k}) = \frac{1}{\sigma} \left(-\frac{c_g}{c_p} \hat{\mathbf{k}} \cdot \mathbb{D} \hat{\mathbf{k}} + \frac{1}{\sigma} \frac{\partial \Omega}{\partial g^*} \frac{Dg^*}{Dt} \right) \quad (20)$$

Waves with a large positive G value will experience rapid exponential growth, and waves with negative G will quickly lose energy. Let us briefly build some intuition for the different terms in G :

- The multiplicative factors $\frac{1}{\sigma}$ and $\frac{c_g}{c_p}$ are wavelength-dependent non-negative functions which scale the energy growth rate. Because of these terms, some wavelengths will grow more quickly than others.
- The $\frac{\partial \Omega}{\partial g^*} \frac{Dg^*}{Dt}$ term says that the energy will grow when this part of the surface experiences a transition from small g^* to large g^* . This happens, for example, at the bottom of a waterfall when ballistic water ($g^* = 0$) collides with a static pool ($g^* > 0$).
- The $-\hat{\mathbf{k}} \cdot \mathbb{D} \hat{\mathbf{k}}$ term describes how the underlying velocity field \mathbf{U} transfers energy to the surface waves. The matrix $-\mathbb{D}$ is positive definite where the velocity field squishes the surface together and negative definite where it stretches the surface apart. The $\hat{\mathbf{k}}$ terms on either side of this matrix mean that the growth rate will have different values depending on the wave’s orientation. Waves aligned with the velocity gradient grow quickly, while waves perpendicular to the velocity gradient do not grow at all. This term is also responsible for the characteristic striped wave textures on stretching surfaces like water falls — the negative definite $-\mathbb{D}$ matrix rapidly decays all waves except the ones perpendicular to the stretched direction, leaving pattern of aligned ripples.

We note that these growth and decay effects are common in a number of advected wave equation solvers, not just our approach. The main reason we care about G is that it tells us where we should expect small waves to grow, so we find it useful for seeding new waves in subsection 4.3.

We also note that this linearized growth analysis is limited in a number of ways. It is unable to capture non-linear effects by definition, and it will not identify situational effects like wind, which may place a bias on the orientation and spectrum of waves. The function

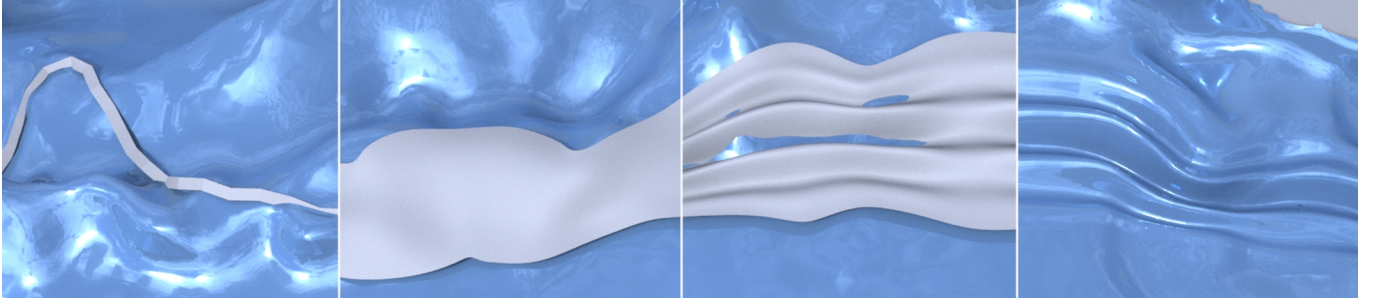


Fig. 5. A single wave curve (far left) approximates the wave behavior within a small neighborhood of itself (middle left). We compute the wave heights by evaluating the phase function and amplitude within the neighborhood of this curve (middle right), and we treat it as a small displacement of the underlying fluid surface (far right). Multiple wave heights are added on top of each other via the superposition principle.

also leads to computational difficulties — finding all waves that maximize this function requires us to evaluate potentially noisy time derivatives and to efficiently sample a four-dimensional function.

4 DISCRETIZATION

Our discretization is heavily inspired by water wave packets [Jeschke and Wojtan 2017]. Instead of representing packets as individual independent particles, we extend the notion of a wave packet to a connected curve restricted to a surface. These curves allow us to better maintain smooth connected wavefronts over long distances, and enable additional modes of artistic control (with curve-editing tools, for example). See Figure 5 for an illustration of the effect of a single wave curve.

4.1 Spatial discretization

We represent a *wave curve* with control points $p_i, i = 1, \dots, n$ and an interpolating function for reconstructing data along the curve. Each control point carries a position \mathbf{x}_i , a wavevector \mathbf{k}_i , a phase θ_i , an action \mathcal{A}_i and a radius r_i , and we use piecewise linear interpolation along the curve. We will use the notation $\mathbf{x}(s, t)$, to indicate the interpolated position at curve parameter s and time t ; we similarly use $\mathbf{k}(s, t)$, $\theta(s, t)$, $\mathcal{A}(s, t)$ and $r(s, t)$ to indicate the interpolated wavevector, phase, action, and radius.

To evaluate the water height (12) at a point \mathbf{y} on the surface, we need to locally reconstruct the phase function θ and the amplitude A from the data stored on the curve. We approximate θ by defining \mathbf{s}_y such that $\mathbf{x}(\mathbf{s}_y, t)$ is the closest point on the curve to the point \mathbf{y} and then using a first order Taylor expansion:

$$\theta(\mathbf{y}, t) \approx \theta(\mathbf{s}_y, t) + \mathbf{k}(\mathbf{s}_y, t) \cdot (\mathbf{y} - \mathbf{x}(\mathbf{s}_y, t)). \quad (21)$$

We reconstruct the amplitude by first computing A_i at each node from the action \mathcal{A}_i using the formula (10). We use the same local kernel function Ψ as wave packets [Jeschke and Wojtan 2017] to describe the falloff of the amplitude away from the curve

$$\Psi(\mathbf{x}, r) = \begin{cases} \frac{1}{2} (\cos(2\pi \frac{\mathbf{x}}{r}) + 1) & \text{if } |\mathbf{x}| \leq r \\ 0 & \text{otherwise.} \end{cases} \quad (22)$$

The amplitude is approximated as

$$A(\mathbf{y}, t) \approx A(\mathbf{s}_y, t) \Psi(\text{dist}(\mathbf{x}(\mathbf{s}_y, t), \mathbf{y}), r(\mathbf{s}_y, t)), \quad (23)$$

where $\text{dist}(\mathbf{x}, \mathbf{y})$ is a function measuring the distance between points \mathbf{x} and \mathbf{y} . Once we have the phase function $\theta(\mathbf{y}, t)$ and the amplitude $A(\mathbf{y}, t)$, we can evaluate the water height $\eta(\mathbf{y}, t) = A(\mathbf{y}, t) \cos \theta(\mathbf{y}, t)$.

4.2 Time discretization

To update \mathbf{x}_i , \mathbf{k}_i , and θ_i we time step equations (15), (16), and (17) respectively with forward Euler integration. To keep the point \mathbf{x}_i on the surface and the wavevector \mathbf{k}_i tangent to the surface, we simply project them onto the surface and the tangent plane after each time step.

The radius r_i has to change based on how the waves get stretched or squashed by the flow. We model this stretching by advecting in the radial direction:

$$\frac{dr}{dt} = \mathbf{n}_i \cdot \nabla(\mathbf{U} + \mathbf{c}_g) \quad (24)$$

where \mathbf{n}_i is the normal vector of the curve lying tangent the surface at point control point i . Unfortunately, the accurate evaluation of $\nabla \mathbf{c}_g$ requires the computation of $\nabla \mathbf{k}$, which is a computationally expensive operation (requiring a neighborhood search of all nearby wave curves) which is not worth the effort in our experience. Instead, we assume $\nabla \mathbf{c}_g$ is negligible compared to $\nabla \mathbf{U}$ and approximate the change in radius according to the background flow only. To ensure that radii do not grow too large or invert due to numerical noise in $\nabla \mathbf{U}$, we clamp r_i to a user-defined range.

To update the wave action \mathcal{A}_i we use the fact that the wave action is conserved per advected area, as discussed in Section 3.4. We associate a trapezoidal area patch to each curve segment and distribute the area a_i to each vertex, as illustrated in Figure 6. In accordance with (18), we keep the area integral of action constant throughout time by setting

$$\mathcal{A}_i^{n+1} = \mathcal{A}_i^n \frac{a_i^n}{a_i^{n+1}}. \quad (25)$$

We use the Lagrangian damping model from Jeschke and Wojtan [2017, Section 3.4] to simulate viscosity and surface contamination effects. This model effectively creates a gradual exponential decay of the packet's amplitude with a rate dependent on its wavenumber.

As waves travel along the surface, their control points can drift away from their ideal spacing over time. To address this we re-sample the wave curves every time step using the polyline re-sampling tools in the *Houdini* software by SideFX, targeting a user-specified spacing between each control point. At the end of this process, we assign the control point data (\mathbf{k} , θ , \mathcal{A} , and r) by interpolating it from the original curve points. We interpolate θ , \mathcal{A} , and r like scalar data; to avoid vector interpolation errors when re-sampling \mathbf{k} , we approximate spherical interpolation by decomposing it into its length k and direction $\hat{\mathbf{k}}$, interpolate them separately, re-normalize $\hat{\mathbf{k}}$, and then re-compose \mathbf{k} . We also delete any wave curve control points with a steepness ratio A/λ less than a minimum threshold (0.01 in our examples).

Over the course of a simulation, wave curves will overlap each other and themselves. The result is unaffected as the linear PDE allows superposition of waves. However, when wave curves fold over themselves *within a single segment*, the inverted region can create noisy caustic waves with large amplitudes. We eliminate this artifact by detecting areas where the wave curve's surface normal points in the opposite direction of the surface normal, and locally setting the wave curve's amplitude to zero. When this algorithm was used in a production environment, effects artists added additional tools for further controlling wave curves, like bounding the maximum steepness. We offer a list of these practical heuristics in Appendix C.

We implemented this wave curve propagation algorithm on two different types of surface geometry (level sets and triangle meshes), and we did not experience any robustness issues. Our naïve surface projection for approximating geodesics requires small enough time steps to avoid waves jumping off of highly-curved surfaces, but it also fails gracefully — if a wave experiences erroneously large deformation or displacement, conservation of wave action naturally kills the amplitude of highly deformed wave packets (just like [Jeschke 2017]). Higher order geodesic-tracing approaches should be possible, but we did not investigate them this paper.

4.3 Discretized wave generation

To generate new wave curves from an existing free-surface fluid flow, we use the theory in Section 3.5. The wave energy growth rate $G(\mathbf{x}, \mathbf{k})$ in Equation 20 will have large values for locations \mathbf{x} and travel directions $\hat{\mathbf{k}}$ that start small and quickly grow in size, so we use this function as an indicator of where to seed new waves. To evaluate this function numerically, we use the base fluid simulation velocity to calculate terms involving \mathbf{U} and its symmetric gradient \mathbb{D} , and we utilize the Navier-Stokes equation to evaluate the effective

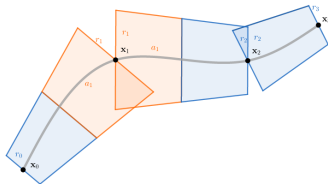


Fig. 6. Area associated with a control point on the wave curve.

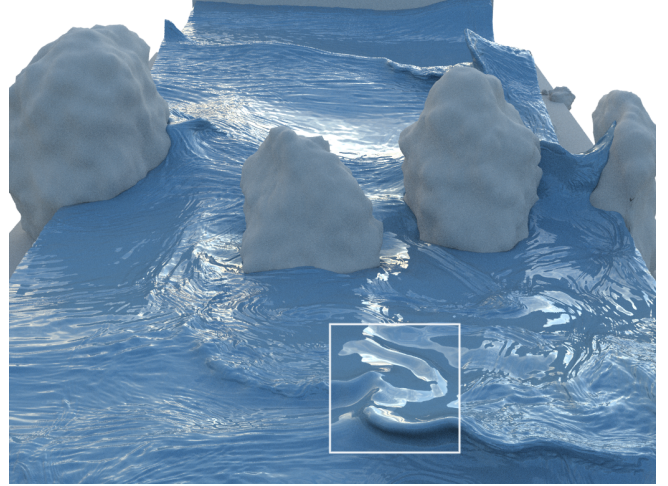


Fig. 7. Wave curves enhance a 3D simulation of waves breaking over rocks. The detail from the underlying fluid simulation is seen in the white box.

gravity in terms of the pressure gradient:

$$\begin{aligned} g^*(\mathbf{x}, t) &= -\mathbf{N}(\mathbf{x}, t) \cdot (\mathbf{g} - \mathbf{a}(\mathbf{x}, t)) \\ &= -\mathbf{N}(\mathbf{x}, t) \cdot \nabla p(\mathbf{x}, t) \end{aligned} \quad (26)$$

To reduce the complexity of sampling and maximizing a four-dimensional function, we heuristically limit our search to a 2D function over space by first pinning down an arbitrary candidate wavenumber k_0 (reducing the 4D $G(\mathbf{x}, k, \hat{\mathbf{k}})$ to a 3D $G(\mathbf{x}, k_0, \hat{\mathbf{k}})$ and then numerically integrating the energy behavior over all wave directions $\hat{\mathbf{k}}$. We then clamp negative values to zero (because we are interested in energy growth rather than decay) and evaluate the resulting 2D function $G(\mathbf{x})$ at all points \mathbf{x} on the fluid surface. We then use $G(\mathbf{x})$ as a density function for Houdini's point-sampling algorithm to get a number of wave seed locations $\tilde{\mathbf{x}}$. At each of these locations, we decide which wave direction should be sampled by finding the $\hat{\mathbf{k}}$ which maximizes the $\hat{\mathbf{k}} \cdot \mathbb{D} \hat{\mathbf{k}}$ stretching term in Equation 20 — the fastest growing $\hat{\mathbf{k}}$ is thus the eigenvector associated with the largest negative value of \mathbb{D} . We now have wave seed positions $\tilde{\mathbf{x}}$ and directions $\hat{\mathbf{k}}$, and we can add multiple wave frequencies by sampling a number of wavenumbers k for each wave seed. These heuristics give us a set of (\mathbf{x}, \mathbf{k}) pairs which should be growing more quickly than most other candidate waves. Note that $G(\mathbf{x}, \mathbf{k})$ describes all wavelengths which are growing, including waves in the background simulation. We avoid double-counting energy by only seeding new waves which are higher frequency than the base simulation.

To generate new wave *curves*, we use the new seed position as a starting point and grow new curves by marching outward in a direction $\dot{\mathbf{x}}$ which is perpendicular to $\hat{\mathbf{k}}$:

$$\dot{\mathbf{x}} = \mathbf{N} \times \hat{\mathbf{k}} \quad (27)$$

To ensure that the wave bends along the surface in a physically appropriate manner (instead of just tracing geodesics), we also update the optimal $\hat{\mathbf{k}}$ as we march. We blend the old $\hat{\mathbf{k}}$ with the

Table 1. Performance timings in seconds per frame

Scene		Base Sim	Wave Curve Sim				Wave Curve Rendering				Total
			Curve points	Emission	Evolution	Total	Samples	Wave stripes	Projection	Total	
breaking wave	Fig. 7	17s	1.5 mil	4.0s	9.2s	13.2s	1 mil	48.3s	74.4s	123s	153s
canal	Fig 2	42s	280k	1.8s	2.0s	3.8s	300k	9.5s	11.7s	21s	67s
paddle	Fig 4	11s	1.2 mil	1.2s	7.1s	8.3s	470k	52.5s	97.8s	150s	170s
paddle($\sigma = ck$)	Fig 11	11s	850k	0.9s	4.6s	5.5s	470k	37.3s	64.5s	102s	118s
river	Fig 3	31s	1 mil	0.9s	5.8s	6.7s	630k	44.8s	39.6s	84s	122s

new one in order to mix in some noise along with the optimal wave directions:

$$\dot{\mathbf{k}} = -\alpha \left(\mathbb{I} - \hat{\mathbf{k}} \otimes \hat{\mathbf{k}} \right) \mathbb{D} \hat{\mathbf{k}} \quad (28)$$

Our examples all use a heuristic blending weight of $\alpha = 2$ ($\alpha = \infty$ creates optimal wave directions, and $\alpha = 0$ creates geodesic wave curves).

Each of these new wave curves is initialized with a radius r (set to 20cm for all waves in our examples) and an amplitude A . We initially set A for each wave curve point to zero, and then we gradually increase its energy over the next 0.5 seconds by $\beta(k)G(\mathbf{x}, \mathbf{k})\Delta t$ each time step. The parameter $\beta(k)$ is a tunable function that we interpret as the wave spectrum of the invisibly small waves discussed in subsection 3.5. We use $\beta(k) = 3.6/k$ in all of our examples.

We are satisfied and encouraged by this new approach to seeding wave energy on a fluid surface, but there is room for improvement. In addition to theoretical limitations discussed in Section 3.5, the evaluation of G will only be as accurate as the base fluid simulation — terms involving the time derivative of the pressure gradient will be particularly noisy for the incompressible fluid simulators typically used in computer animation. Furthermore, we found if we *only* sample the waves which optimize G , then the resulting water surfaces can appear too orderly and sterile. To seed a light amount of isotropic random noise across the surface, our examples add a small constant term to G . The subsequent physical evolution of the curves will naturally damp out less appropriate waves and amplify dominant ones.

4.4 Rendering

To visualize waves carried by wave curves, we generate a high resolution surface capturing all the fine detail. Steps of this process are roughly visualized in Figure 5.

As the first step, every wave curve is turned into a *wave stripe*, a thickened wave curve with width given by the radius r carried by each wave curve point. Afterward, the displacement information is projected onto the high resolution surface. We send a ray in the normal direction from every vertex of the high resolution surface, and every time a ray intersects a *wave stripe* we use Equations 21 and 23 to compute the phase function θ_i and the amplitude A_i . Each vertex on the high resolution surface is then displaced in the normal direction by $\sum_i A_i \sin \theta_i$. However, when many wave curves are at a single place, this total displacement can become excessively large.

The visual impact of a wave curve is mainly given by the wave steepness $s_i = A_i k_i$, and we found the total steepness $s_t = \sum_i s_i$ to be a good predictor of large wave displacements. Based on our

experiments, we smoothly limit wave steepnesses to a critical value $s_c = 3$, giving us the final displacement η :

$$\eta = \frac{s_c}{s_t} \tanh \frac{s_t}{s_c} \sum_i A_i \sin \theta_i. \quad (29)$$

The factor $\frac{s_c}{s_t} \tanh \frac{s_t}{s_c}$ can be interpreted as a type of soft-clamping operation: it is close to one if the total steepness s_t is small, but it appropriately scales down the displacement for large s_t . This scaling based on total steepness seems to work nicely across different wavelengths.

5 RESULTS

We implemented our method on top of a FLIP-based fluid simulation [Bridson 2015]. We started by pre-computing several liquid simulations: a canal with obstacles (Figure 2), waves breaking over rocks (Figure 7), a large river (Figure 3), and the churning paddle *Houdini* scene from [Lait 2011] (Figure 4). We then simulated additional high-frequency details on top of these dynamic fluid surfaces as a post-process. We use the algorithm in subsection 4.3 to seed new wave curves each time step for wavelengths $\lambda = 20\text{cm}, 10\text{cm}, 5\text{cm}, 2.5\text{cm}$, and 1.25cm . This strategy requires the simulations to run for a period of time before the wave curves “saturate” the scene.

We report a number of interesting wave effects visible in our simulations. The advection of the wave curves creates streaks as they are swept away with the current, and the dispersive dynamics cause different wavelengths to travel at different speeds. Standing waves occasionally appear where waves travel upstream at the exact same rate as the background flow. Most importantly, our simulated waves are independent of the simulation resolution, so we can move the camera close to a fluid simulation without running out of observable wave details.

We implemented our algorithm in Houdini using its scripting language VEX, and we measured computation times on an Intel Core i7-7820X 3.60GHz CPU with 8 cores. Table 1 and 2 show the time our algorithm took to simulate wave curves on top of each simulation, as well as the time to generate the high resolution surface used for rendering. Each simulation time step consists of four operations: the evolution of the wave curves (Sections 4.1 and 4.2), the generation of new wave curves (Section 4.3), the construction of wave stripes and projection onto the high resolution surface (Section 4.4). Table 2 shows times for running the wave curves simulation with a different number of wave curves. Thanks to the embarrassingly parallel nature of our algorithm we can see that the time scales linearly with the number of wave curve points. The exception is the wave curve

Table 2. Performance timings (seconds per frame) for different numbers of wave curves (Figure 8).

Curves points	Emission	Wave evolution	Wave stripes	Projection	Total
100k	0.60s	0.72s	4.49s	5.42s	11.22s
200k	0.58s	1.32s	9.43s	11.18s	22.50s
400k	0.64s	2.50s	18.89s	23.69s	45.72s
800k	0.78s	4.25s	32.50s	47.68s	85.21s

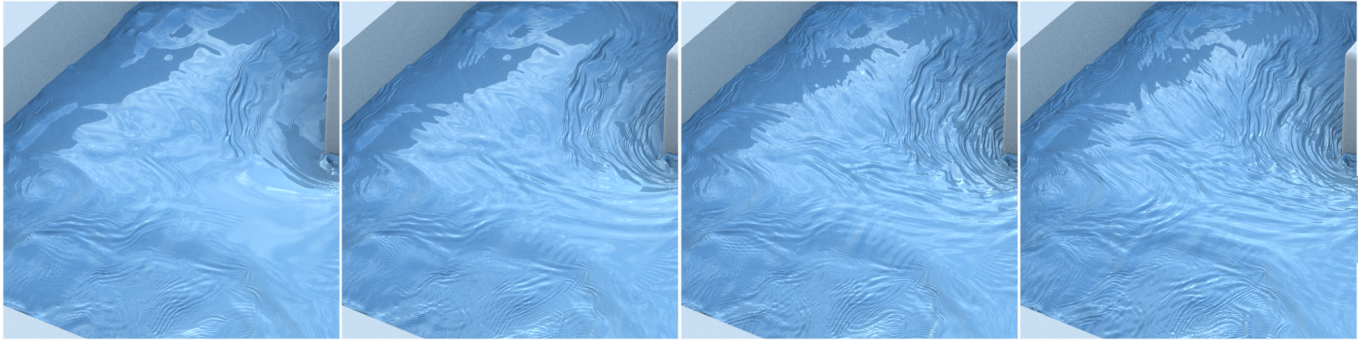


Fig. 8. Difference between simulations with 100,000(left), 200,000, 400,000 and 800,000 (right) wave curve points.

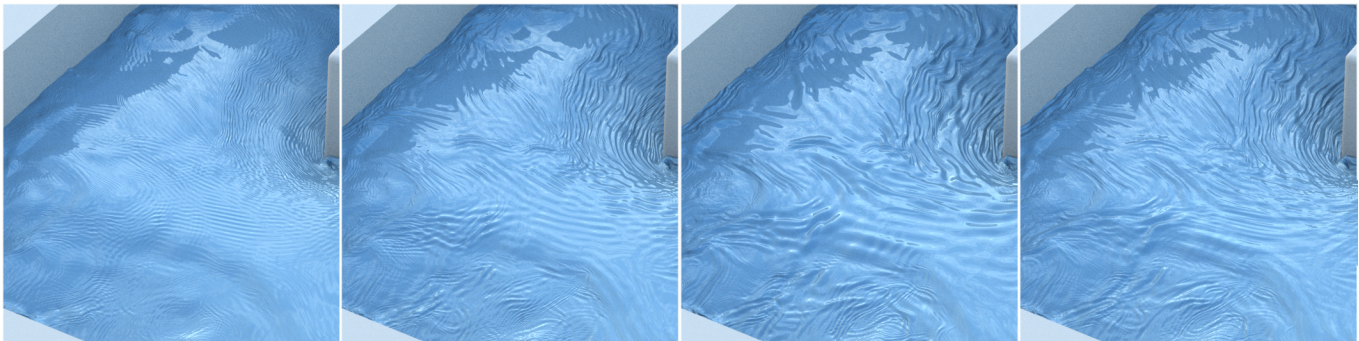


Fig. 9. Difference between simulations where only waves of a single wavelength are generated, 5cm(left), 10cm, 20cm. The right most image show a simulation of all five wavelengths simultaneously.

generation step where the cost is mainly dictated by the complexity of the base simulation.

Figure 8 shows the effect of adding a variable number of wave curves on top of a simulation. As expected, more wave curves make the simulation seem more detailed. Using very few curves tends to leave open expanses without any ripples, and the contrast between high resolution wave detail and a complete absence of detail can look unnatural. Our wave curves also create long, connected wavefronts, which biases the wave spectrum toward coherent waves. If a perfectly isotropic wave spectrum is undersampled by our method, then the spectrum will consist of few waves in a few randomly chosen directions, which may also appear unnatural.

Figure 9 shows the effect of using only a single wavelength instead of a spectrum of several wavelengths in our simulations. The use of a large spectrum of wavenumbers clearly influences both the visual details as well as the physical plausibility of the animation.

Figure 10 compares our approach to the method of Kim et al. [2013]. We note that the simulation from Kim et al. [2013] creates more localized high frequency waves near splashes on this example, while our seeding function distributes wave detail throughout the scene and across a wider range of frequencies. Our approach produces many high-frequency wave features due to its independence from any grid resolution, but our damping model makes the shortest waves rapidly decay.

Figure 11 compares to a simulation with a much simpler dispersion relation. Instead of the wavelength-dependent speeds described in subsection 3.1, all wavelengths in this animation move at a constant speed. In contrast to Figure 9, the differences in this one are admittedly subtle; we observe disparities in the wave spectrum at different locations, but we find it difficult to declare which one looks more “realistic.” We tentatively conclude from these two experiments that the range of simulated wavelengths seems more

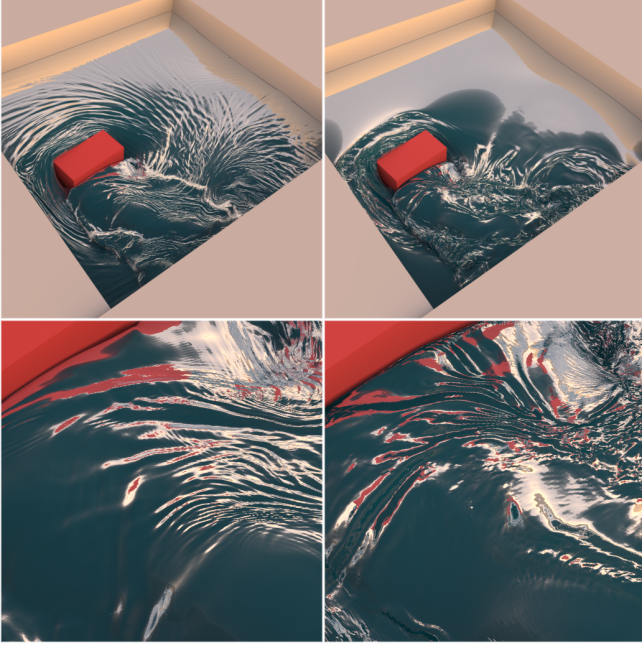


Fig. 10. A comparison between our wave curve approach (left) and the method of Kim et al. [2013] (right) for the same background simulation.

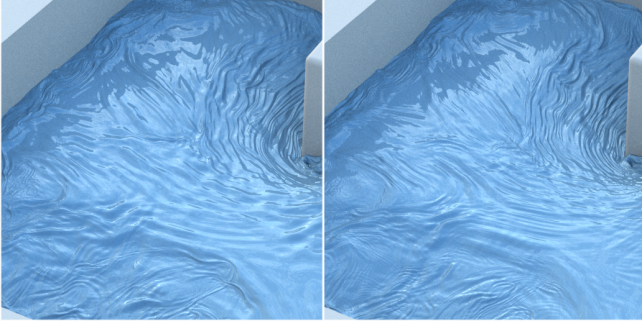


Fig. 11. We only observe subtle differences between a simulation with a simple dispersion relation $\sigma = ck$ (left), and a simulation with our dispersion relation for water waves (right).

important than accurate propagation velocities for the purpose of creating realistic liquid animations for visual effects.

On the other hand, we do believe the physically-derived dispersion relation is useful for the purpose of *seeding* waves. Figure 12 compares the wave seeding algorithm in section subsection 4.3 to a simpler technique which places waves in random locations and orientations each time step. We chose the seeding rate for the random approach such that it would produce approximately the same number of wave curve points as our approach by the end of the simulation. Our seeding algorithm seems to produce waves that align with fluid surface features and grow to a sizable amplitude, while the random approach scatters small waves across the surface without a coherent wave direction emerging until after a great many

waves have been created. Our energy-based seeding approach also seems to be more efficient: the number of wave curves are about the same 100 timesteps into the animation (Figure 13 top), despite the fact that our approach spawned far fewer waves during that time period (Figure 13 bottom).

6 LIMITATIONS & DISCUSSION

We reiterate here that the theory in subsection 3.2 assumes small wave amplitudes relative to their wavelength, a slowly varying environment, and small wavelengths relative to the overall fluid simulation, and we should expect to see inaccuracies when these assumptions are violated. Indeed, large amplitudes will cause waves to propagate at the wrong speed and will look unrealistically choppy, and extremely long wavelengths will cause our waves to unnaturally compete with features in the base fluid simulation. If our wave curve discretization is too coarse relative to the base simulation, then

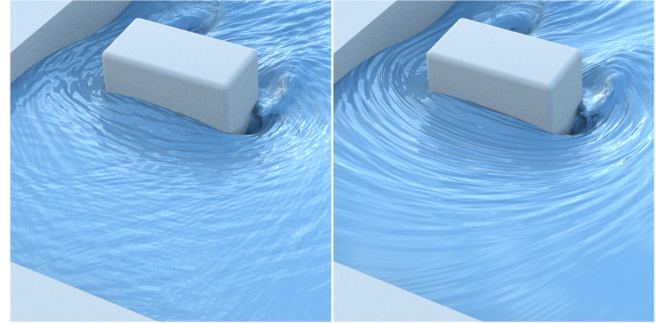


Fig. 12. A simulation with random seeding (left) and our proposed seeding strategy based on energy growth rates (right). The random approach generates a noisy collection of many small waves throughout the surface. Our approach exhibits coherent waves aligned with the underlying fluid motion.

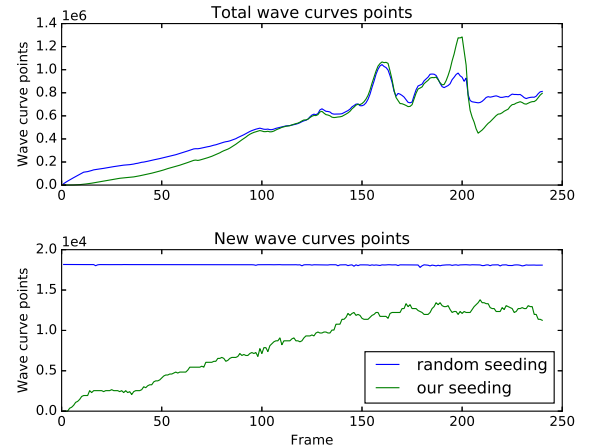


Fig. 13. Number of total and new wave curves points at each frame in the paddle scene (Figure 12). Sudden peaks in the total number of wave curve points are caused by the underlying fluid motion inducing stretching and subdivision of the wave curves.

waves produced by the same curve segment will behave the same even if they are located in very distant places.

Additionally, the time integration procedure in subsection 4.2 is effectively raytracing, which again assumes a small wavelength relative to the environment and does not model diffraction without introducing additional techniques like [Jeschke and Wojtan 2015]. Curve tracing can produce caustics, and currents which exactly match the wave speed (wave blocking) can theoretically pile up wave heights at a particular location. We resolve these large-amplitude violations by thresholding the total wave steepness, as discussed in Appendix C.



Section 3.3 introduces the effective gravity g^* as a result of a non-inertial reference frame. It is not clear to us whether rotational fictitious forces (coriolis, centrifugal, Euler) should also appear in this theory, or whether they are ultimately negligible. Related work in the fluid dynamics literature [Longuet-Higgins 1995] also neglects these effects, but we have not yet seen a convincing proof either way.

We implemented our wave curve algorithm into a visual effects production pipeline. Aside from the benefits of increased visual resolution, the simulation artists reported additional enthusiasm for the controllability of our approach. Specifically, the Lagrangian curve primitives make it easy to manually override the amplitude and motion of any individual curve in a scene.

7 CONCLUSION

We have presented an algorithm for incorporating highly detailed water waves onto a moving liquid surface. We derive the relevant equations of motion for Lagrangian wave curves and introduce a novel wave seeding criterion. The resulting method efficiently enhances the surface detail of a pre-computed water animation, and the final visual resolution is independent of the simulation's computational cost.

ACKNOWLEDGMENTS

We wish to thank the anonymous reviewers and the members of the Visual Computing Group at IST Austria for their valuable feedback. This research was supported by the Scientific Service Units (SSU) of IST Austria   through resources provided by Scientific Computing. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 638176.

REFERENCES

- D. J. Acheson. 1990. *Elementary fluid dynamics*. Clarendon Press Oxford University Press, Oxford New York.
- George Biddell Airy. 1841. Tides and waves. (1841).
- Roland Angst, Nils Thuerey, Mario Botsch, and Markus Gross. 2008. Robust and Efficient Wave Simulations on Deforming Meshes. *Computer Graphics Forum* 27, 7 (Oct 2008), 1895–1900.
- Omri Azencot, Orestis Vantzos, and Mirela Ben-Chen. 2018. An explicit structure-preserving numerical scheme for EPDiff. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 107–119.
- Morten Bojsen-Hansen, Hao Li, and Chris Wojtan. 2012. Tracking surfaces with evolving topology. *ACM Trans. Graph.* 31, 4 (2012), 53–1.
- Morten Bojsen-Hansen and Chris Wojtan. 2013. Liquid surface tracking with error compensation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 68.

- Robert Bridson. 2015. *Fluid Simulation for Computer Graphics, Second Edition*. A K Peters/CRC Press.
- José A. Canabal, David Miraut, Nils Thuerey, Theodore Kim, Javier Portilla, and Miguel A. Otaduy. 2016. Dispersion Kernels for Water Wave Simulation. *ACM Trans. Graph.* 35, 6, Article Article 202 (Nov. 2016), 10 pages.
- Hilko Cords. 2008. Moving with the Flow: Wave Particles in Flowing Liquids. *Journal of WSCG* 16, 1-3 (2008), 145–152.
- P. G. Drazin. 2002. *Introduction to Hydrodynamic Stability*. Cambridge Univ. Press.
- Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. 2006. Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques (SIGGRAPH '06). Association for Computing Machinery, New York, NY, USA, 805–811.
- Stefan Jeschke, Tomáš Skřivan, Matthias Müller-Fischer, Nuttapon Chentanez, Miles Macklin, and Chris Wojtan. 2018. Water surface wavelets. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 94.
- Stefan Jeschke and Chris Wojtan. 2015. Water Wave Animation via Wavefront Parameter Interpolation. *ACM Trans. Graph.* 34, 3, Article Article 27 (May 2015), 14 pages.
- Stefan Jeschke and Chris Wojtan. 2017. Water Wave Packets. *ACM Trans. Graph.* 36, 4, Article Article 103 (July 2017), 12 pages.
- Michael Kass and Gavin Miller. 1990. Rapid, stable fluid dynamics for computer graphics. *ACM SIGGRAPH Computer Graphics* 24, 4 (Sep 1990), 49–57.
- Theodore Kim, Jerry Tessendorf, and Nils Thürey. 2013. Closest Point Turbulence for Liquid Surfaces. *ACM Trans. Graph.* 32, 2, Article Article 15 (April 2013), 13 pages.
- Jeff Lait. 2011. Correcting low frequency impulses in distributed simulations. In *ACM SIGGRAPH 2011 Talks*. 1–2.
- MS Longuet-Higgins. 1985. Accelerations in steep gravity waves. *Journal of physical oceanography* 15, 11 (1985), 1570–1579.
- Michael S. Longuet-Higgins. 1995. Parasitic capillary waves: a direct calculation. *Journal of Fluid Mechanics* 301, -1 (Oct 1995), 79.
- Jörn Loviscach. 2002. A Convolution-Based Algorithm for Animated Water Waves.. In *Eurographics (Short Papers)*.
- Chiang Mei. 2005. *Theory and applications of ocean surface waves*. World Scientific, Singapore Hackensack, NJ.
- Olivier Mercier, Cynthia Beauchemin, Nils Thuerey, Theodore Kim, and Derek Nowrouzezahrai. 2015. Surface turbulence for particle-based liquid simulations. *ACM Transactions on Graphics* 34, 6 (Oct 2015), 1–10.
- Björn Ottosson. 2011. *Real-time interactive water waves*. Ph.D. Dissertation. Master's thesis, KTH.
- Sanjit Patel, Jerry Tessendorf, and Jeroen Molemaker. 2009. Monocoupled 3D and 2D river simulations. In *Proc. ACM/Eurographics Symp. Comp. Anim., Posters Session*.
- Jerry Tessendorf. 2002. Simulating Ocean Water. (01 2002).
- Jerry Tessendorf. 2004. Interactive Water Surfaces. (2004), 265–274. https://people.cs.clemson.edu/~jtessen/papers_files/Interactive_Water_Surfaces.pdf
- Nils Thürey, Ulrich Rüde, and Marc Stamminger. 2006. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 157–164.
- Nils Thürey, Chris Wojtan, Markus Gross, and Greg Turk. 2010. A Multiscale Approach to Mesh-Based Surface Tension Flows. *ACM Trans. Graph.* 29, 4, Article Article 48 (July 2010), 10 pages.
- Huamin Wang, Gavin Miller, and Greg Turk. 2007. Solving General Shallow Wave Equations on Surfaces (SCA '07). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 229–238.
- G. B. Whitham. 1999. *Linear and nonlinear waves*. Wiley, New York.
- Sheng Yang, Xiaowei He, Huamin Wang, Sheng Li, Guoping Wang, Enhua Wu, and Kun Zhou. 2016. Enriching SPH Simulation by Approximate Capillary Waves (SCA '16). Eurographics Association, Goslar, DEU, 29–36.
- Jihun Yu, Chris Wojtan, Greg Turk, and Chee Yap. 2012. Explicit Mesh Surfaces for Particle Based Fluids. *Computer Graphics Forum* 31, 2pt4 (May 2012), 815–824.
- Cem Yuksel, Donald H. House, and John Keyser. 2007. Wave Particles. *ACM Trans. Graph.* 26, 3, Article 99 (July 2007).

A LAGRANGIAN EVOLUTION

Here we show that dispersion relation (14) defines the evolution of waves. Combining Equations 8 and 14 gives us

$$-\frac{\partial \theta}{\partial t} - \mathbf{U} \cdot \nabla \theta = \Omega(\|\nabla \theta\|, g^*). \quad (30)$$

Rewriting this reveals a Hamiltonian structure

$$-\frac{\partial \theta}{\partial t}(\mathbf{x}, t) = H(\mathbf{x}, \nabla \theta(\mathbf{x}, t), t), \quad (31)$$

$$H(\mathbf{x}, \mathbf{k}, t) = \Omega(\|\mathbf{k}\|, g^*(\mathbf{x}, t)) + \mathbf{U}(\mathbf{x}, t) \cdot \mathbf{k}. \quad (32)$$

with a phase space (\mathbf{x}, \mathbf{k}) in place of the familiar (q, p) . This structure allows us to interpret the dispersion relation (14) as a Hamilton-Jacobi equation, which has a couple of interesting consequences. For example, note that the wavevector \mathbf{k} plays the role of the momentum of a wave. Furthermore, because a Hamiltonian is constant along trajectories if it is time independent, the absolute frequency ω (which is equal to our Hamiltonian) is constant as long as $g^*(\mathbf{x}, t)$ and $\mathbf{U}(\mathbf{x}, t)$ do not change in time.

We can now derive Hamiltonian equations of motion:

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \mathbf{k}} = \frac{\partial \Omega}{\partial \mathbf{k}} \hat{\mathbf{k}} + \mathbf{U}, \quad (33)$$

$$\dot{\mathbf{k}} = -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial \Omega}{\partial g^*} \nabla g^* - [\nabla \mathbf{U}]^T \mathbf{k}. \quad (34)$$

which tell us that the wave position changes with the mean velocity \mathbf{U} and group velocity $\mathbf{c}_g = \frac{\partial \Omega}{\partial \mathbf{k}} \hat{\mathbf{k}}$, and that the wavevector changes with variations in effective gravity g^* and mean velocity \mathbf{U} .

We can also derive want the evolution of the phase function along the trajectory. Adding $-\mathbf{c}_g \cdot \nabla \theta$ to both sides of (30) allows us to complete the chain rule for $\dot{\theta}$, giving us

$$\dot{\theta} = -\Omega(k, g^*) + \mathbf{c}_g \cdot \mathbf{k}. \quad (35)$$

B THE BALANCE OF ENERGY

The balance of energy can be obtained from the conservation of the wave action by starting with the definition of action $\mathcal{A} = \frac{E}{\sigma}$ and rewriting it into the form of a balance law.

$$0 = \frac{\partial}{\partial t} \frac{E}{\sigma} + \text{div} \left((\mathbf{U} + \mathbf{c}_g) \frac{E}{\sigma} \right) = \frac{\partial E}{\partial t} + \text{div} ((\mathbf{U} + \mathbf{c}_g) E) \quad (36)$$

$$- \frac{E}{\sigma^2} \left(\frac{\partial \sigma}{\partial t} + (\mathbf{U} + \mathbf{c}_g) \cdot \nabla \sigma \right) \quad (37)$$

The right hand side is basically in the form we want, but we will massage the last term into something a little easier to interpret. In the light of Appendix A, the dot-time derivative of some quantity X is $\dot{X} = \frac{\partial X}{\partial t} + (\mathbf{U} + \mathbf{c}_g) \cdot \nabla X$. Combined with $\sigma = \Omega(k, g^*)$ we obtain

$$\left(\frac{\partial \sigma}{\partial t} + (\mathbf{U} + \mathbf{c}_g) \cdot \nabla \sigma \right) = \dot{\sigma} = \dot{\Omega}(k, g^*) \quad (38)$$

$$= \frac{\partial \Omega}{\partial \mathbf{k}} \cdot \dot{\mathbf{k}} + \frac{\partial \Omega}{\partial g^*} \dot{g}^*, \quad (39)$$

Using the evolution (34) of \mathbf{k} and identities $\mathbf{c}_g = \frac{\partial \Omega}{\partial \mathbf{k}}$, $\dot{g}^* = \frac{Dg^*}{Dt} + \mathbf{c}_g \cdot \nabla g^*$ yields

$$\frac{\partial \Omega}{\partial \mathbf{k}} \dot{\mathbf{k}} + \frac{\partial \Omega}{\partial g^*} \dot{g}^* = -\mathbf{c}_g \cdot [\nabla \mathbf{U}]^T \mathbf{k} + \frac{\partial \Omega}{\partial g^*} \frac{Dg^*}{Dt} \quad (40)$$

$$= -k c_g \hat{\mathbf{k}} \cdot \mathbb{D} \hat{\mathbf{k}} + \frac{\partial \Omega}{\partial g^*} \frac{Dg^*}{Dt}, \quad (41)$$

where $\mathbb{D} = \frac{1}{2} (\nabla \mathbf{U} + [\nabla \mathbf{U}]^T)$. Putting everything together yields the final balance of energy

$$\frac{\partial E}{\partial t} + \text{div} ((\mathbf{U} + \mathbf{c}_g) E) = \frac{E}{\sigma} \left(-\frac{c_g}{c_p} \hat{\mathbf{k}} \cdot \mathbb{D} \hat{\mathbf{k}} + \frac{1}{\sigma} \frac{\partial \Omega}{\partial g^*} \frac{Dg^*}{Dt} \right). \quad (42)$$

C PRACTICAL WAVE CURVE HEURISTICS

Linear wave theory has its limitations and breaks down in certain scenarios. The main assumption is that the steepness, $s = Ak$, is small. We enforce maximum steepness of 0.7 by clamping wave amplitudes when they get too large, as suggested by [Jeschke and Wojtan 2015].

Furthermore, it does not make much sense to have a large wave curve on small droplets and splashes, since a large wave would rip these into even smaller droplets and splashes in reality. To roughly detect these cases, we compute the maximal curvature κ_{max} of the base simulation surface and clamp the wave curve amplitude A and radius r such that $A\kappa_{max} < 0.5$ and $r\kappa_{max} < 0.3$.

We also found it practical to enforce a maximum number of wave curves points for each simulation, usually around one million. To remove excess wave curves each time step, we mark points with the smallest steepness that are above the limit, gradually reduce their amplitude to zero over the next five time steps, and then remove from the simulation. Because this procedure would have the negative side effect of also deleting newly generated wave curves, we exclude all points from this deletion procedure during the first ten timesteps of their lifetime.