Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2014)
Vladlen Koltun and Eftychios Sifakis (Editors)

Ocean Waves Animation using Boundary Integral Equations and Explicit Mesh Tracking

T. Keeler and R. Bridson

University of British Columbia

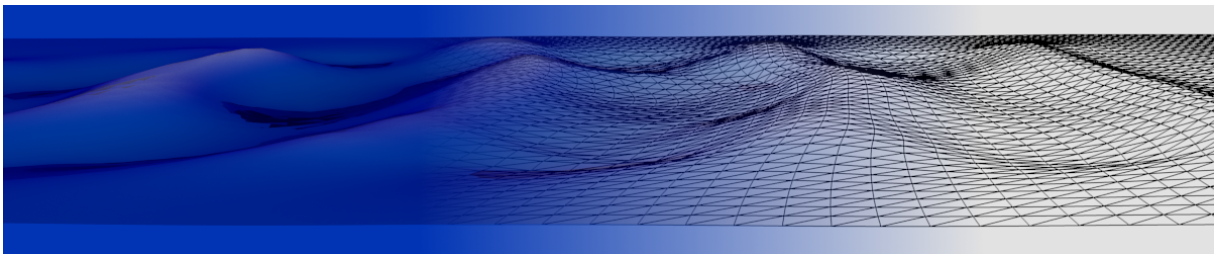

Figure 1: *Deep Ocean Waves*

Abstract

We tackle deep water simulation in a scalable way, solving 3D irrotational flow using only variables stored in a mesh of the surface of the water, in time proportional to the rendered mesh. The heart of our method is a novel boundary integral equation formulation that is amenable to explicit mesh tracking with unstructured triangle meshes. Our method complements FFT style waves as it is able to handle solid boundaries. It is less memory intensive than volumetric methods and inherently handles the near-infinite depth of the deep ocean. We demonstrate acceleration techniques using the FMM and GPU computing. The natural Lagrangian motion of our model gives inherent adaptivity to our simulation without the need for direct mesh operations.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation; Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based modeling;

1. Introduction

Ocean waves are a common animation challenge. To achieve natural-looking and rich detail, physical simulation of one kind or another has generally been adopted. However, there is an inconvenient gap between highly efficient but limited FFT-based surface displacement methods and full 3D simulation of a large volume of water, which we aim to bridge in this paper.

For non-overturning waves without significant interaction with boats or other solids, or just to provide a good ripple-scale animated displacement texture, Tessendorf's extremely efficient FFT-based solver [T*01] is typically used. This model is derived from an irrotational flow assumption (i.e. that the flow beneath the surface has negligible vorticity) which has been experimentally validated for gentle to moderate waves, along with several other simplifications. However, its realism falters for stormy oceans, where the assumption of a small and smooth perturbation from flat geometry is grossly broken, and doesn't account for the presence of large solids in the water.

For full-fledged interaction with solids and for overturning waves with splashes, 3D simulation of the free-surface Navier-Stokes equations is the main alternative. However, this comes with a significant performance cost, as a large volume of water beneath the surface must be discretized and

solved: though it is not directly rendered, the effect of the depth is plainly visible in terms of wave speeds and dispersion. The finite simulated domain generally has to be much smaller than the field of view in rendering as well, requiring nontrivial effort to convincingly blend the 3D region into some sort of continued surface geometry out to the edges of the view.

The core of our contribution consists of expanding on principles laid out in engineering literature that are found in papers such as Xue et al. [XXLY01], Liu et al. [LXY01], and Grilli et al. [GGD01]. In these papers, numerical methods solving wave dynamics rely on boundary integral equations (BIEs) to do surface-only computations to propagate wave dynamics while allowing solid body interaction with submerged and partially-submerged objects.

Our method differs from those referenced in that we use an indirect integral formulation which obviates the need for high-order geometry and function approximation (required by the engineering simulations). As a result we are able to employ explicit mesh tracking using the El Topo library [BB09] to represent the mesh in a robust and adaptive manner.

There also exists an accelerated BEM wave simulation, developed by Fochesato et al. [FD06], using principles based on the Fast Multipole Method (FMM), an acceleration algorithm due to Greengard et al. [GR87]. Our reformulation is immediately amenable to the FMM and we use it to demonstrate the favourable asymptotic complexity of our simulation.

In short, our method presents the following novel contributions to the animation of water wave phenomena:

- a surface-only memory-efficient method for computing deep ocean waves,
- a boundary integral formulation better fitted to coarse explicit meshes, including solid boundaries,
- a method which implicitly satisfies deep water conditions without needing a large simulation domain,
- optimal scaling of computation time using advanced hardware and algorithms.

1.1. Related Work

The use of BIE methods in graphics began with James and Pai [JP99] for elastic deformation. However, the most common use has been in the computation of solid boundaries for vortex methods. Many methods construct sheets of singular vorticity on solid surfaces which then cancel normal flow. Solving for the necessary vortex sheet strength constitutes solving a BIE: see Park and Kim [PK05], Weissman and Pinkall [WP10] and Golas et al. [GNS*12] for this approach to incorporating solid boundaries in vortex-based smoke simulations. Brochu et al. [BKB12] used a different BIE formulation called the single layer potential to compute the potential flow satisfying free-slip boundary conditions necessary in a vortex method, and demonstrated using the FMM combined with iterative methods to accelerate computing the solution.

Wave animation in general has a long history in graphics, and we do not attempt to document it all. Looking at surface models which can be derived from potential flow, Fournier and Reeves' [FR86] and Peachey's [Pea86] physically-based procedural models are a major milestone, using Gerstner or Rankine waves and also including the effect of solid boundaries such as beaches. Tessendorf's FFT simulation [T*01] mentioned above also belongs to the potential flow / surface category. Other researchers have turned to discretizing the shallow water equations, which allows for a very efficient surface-based simulation [LvdP02, ATBG08, CM10] but only approximates shallow waves: the realistic dispersion of deep water ocean waves is not captured. Yuksel's wave particle approach [YHK07] similarly does not capture the dispersion of deep water waves.

Full three-dimensional water solvers can be applied to ocean scenes, though using a volumetric grid deep enough to effectively model deep water waves, broad enough to capture a reasonably large surface area, and detailed enough to get adequate surface resolution can make this extremely expensive. Irving et al. [IGLF06] and Chentanez and Müller [CM11] incorporated vertically stretched grid cells in their solvers to make deep water more economical. Nielsen and Bridson [NB11] guided a high resolution simulation of just the surface layer of the ocean with a coarse but much deeper simulation to capture deep water effects.

2. Deep Water Waves

The key assumption in our non-linear waves formulation is that the wave motion is irrotational and thus can be modelled by potential flow. This assumption is commonly held, and has experimental evidence supporting it. Potential flow is a divergence-free velocity field $\vec{u}$ obtained by taking the gradient of a scalar potential function $\Phi(\vec{x})$, where $\Phi(\vec{x})$ solves Laplace's equation,

$$\nabla^2 \Phi = 0. \tag{1}$$

Potential flow is uniquely defined by its boundary and the conditions thereon. It is natural then to restrict our mathematical and computational model to the boundary using integral equations.

For the deep ocean, as seen in Figure 2, our scalar potential exists in an infinite domain Ω with boundary Γ comprised of solid boundaries Γ_s and the free surface representing the water-air interface Γ_f. We will denote points on the surfaces with subscripts: $\vec{x}_f \in \Gamma_f$, $\vec{x}_s \in \Gamma_s$ and $\vec{x}_\Gamma \in \Gamma$.

On the free surface the solution to Bernoulli's equations

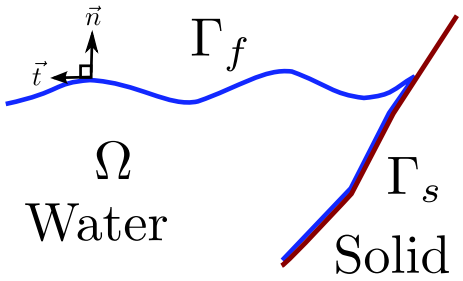

Figure 2: *The geometry of our wave formulation.*

(which govern potential flow) gives a time dependent Dirichlet condition,

$$\frac{D}{Dt}\Phi(\vec{x}_f) = \tfrac{1}{2}|\nabla\Phi(\vec{x}_f)|^2 - g\, z(\vec{x}_f) - p(\vec{x}_f), \qquad (2)$$

for the atmospheric pressure p, height z, gravitational acceleration g, and the convective derivative $D/Dt = \partial/\partial t + (\vec{u}\cdot\nabla)$. At solid boundaries Γ_s in the fluid we impose the normal velocity $\vec{n}\cdot\vec{u}_s$ of the solid as a Neumann boundary condition,

$$\frac{\partial}{\partial n}\Phi(\vec{x}_s) = \vec{n}(\vec{x}_s)\cdot\vec{u}_s(\vec{x}_s). \qquad (3)$$

To approximate the near infinite depth of the ocean floor, we apply the far-field condition $\Phi \to 0$ as $z \to -\infty$.

The wave motion is given by advecting the surface Γ by the gradient of the potential,

$$\frac{D}{Dt}\vec{x}_\Gamma = \nabla\Phi(\vec{x}_\Gamma) \qquad (4)$$

The entire system is determined at a time t by the variables $\Phi(\vec{x}_f)$, $\Gamma(t)$ and solid boundary velocity. The system is initialized by defining these variables at time $t = 0$. We generally operate under the assumption that the atmospheric pressure p is zero on the boundary. Added modelling could include variable pressure such as that due to wind forces.

2.1. Linearization

The non-linear term in Bernouli's equation (2) is measurably small for ocean waves. We moreover find it can be destabilizing if included in the numerics, thus we linearize the equations by dropping the non-linear term, consistent with Tessendorf's commonly-used FFT wave simulations. We are similarly able to simulate the shallow troughs and steep peaks of ocean waves without it, as we follow the full motion of the surface and not just the vertical displacement. We expect in future work that more advanced numerical treatment will allow stable inclusion of this term when it is relevant, allowing for a wider range of animated phenomena where the non-linear term is more important such as wave breaking and overturning waves.

2.2. Boundary Integral Formulation

The potential flow wave formulation above lends itself directly to reformulation using a boundary integral equation. Boundary integrals represent the solution interior by integrating over data contained on its boundaries; finding the solution requires matching the boundary data to the given boundary conditions. Integral equations come in two types, direct and indirect. In *direct* integral equations the boundary data is some physically relevant quantity, such as the solution values or normal derivative on the boundary. The aforementioned engineering literature adopts this approach in their approach to wave modeling, using Green's second identity to represent the interior solution as an integral involving both Dirichlet and Neumann data. For reasons below, we instead take an alternative approach, an *indirect* integral formulation, where boundary data used to represent the potential solution is a fictitious density function σ called a single layer potential,

$$\Phi(\vec{x}) = \int_\Gamma \sigma(s)G(\vec{x},\vec{y}(s))dS, \vec{x} \in \Omega. \qquad (5)$$

The kernel G is the fundamental solution to Laplace's equation in three dimensions, $G(\vec{x},\vec{y}) = \frac{1}{4\pi|\vec{x}-\vec{y}|}$.

In the limit as $\vec{x}$ approaches the boundary, we have the following combined first/second kind integral equation for σ given the respectively Dirichlet/Neumann boundary conditions on Γ:

$$\Phi(\vec{x}_f) = \int_\Gamma \sigma(\vec{y}(k))G(\vec{x}_f,\vec{y}(k))dk, \qquad (6)$$

$$\frac{\partial}{\partial n}\Phi(\vec{x}_s) = \frac{\sigma(\vec{x}_s)}{2} + \frac{\partial}{\partial n_s}\int_\Gamma \sigma(\vec{y}(k))G(\vec{x}_s,\vec{y}(k))dk. \qquad (7)$$

Once σ is calculated, we can evaluate the gradient of Φ on the surface by taking the gradient of (5) and replacing the normal component with (7). This process of computing the gradient is straightforward and lends itself to our numerics. Attempting a similar procedure with the direct formulation leads to hyper-singular integrals, which are far more difficult to handle numerically especially on coarse triangle meshes. Additionally, the indirect formulation sidesteps a numerical oddity of the direct formulation, where the discretization treats the Neumann and Dirichlet data with the same approximation basis.

A primary characteristic of our formulation and those in the engineering literature is that they involve solving a Fredholm integral equation of the first kind (6). Mathematically, this implies that finding σ may be somewhat ill-conditioned since our kernel is weakly singular (roughly, weakly singular means the kernel is singular, but integrable in L^2)

and therefore the integral operator is compact. In practice, both we and the engineering literature find meaningful results from solving such a system, though care must be taken when handling the singular integrals to avoid unsolvable systems, and even then some degree of high-frequency artifacts should be expected and regularized or smoothed away. Regardless, handling both free surface and solid boundaries with straightforward integral equations likely necessitates solving a first kind equation if we want to avoid solving systems with hyper-singular kernels (again roughly, hyper-singular means the kernel singularity is not L^2 integrable and that the integral operator must be considered in a special manner).

Advantages of the boundary integral formulation include:

- Resolving everything on the surface dimensionally reduces our problem.
- Integral equations have some surprising numerical characteristics that further increase this efficiency
- The infinite depth of the ocean is implicitly satisfied by the mathematical formulation.

We will further elaborate on the numerical advantages in the following section.

3. Numerical Method for Boundary Integral Waves

To numerically solve the wave equations we take the following pseudo-algorithm:

1. Solve the BIEs (6) and (7) for σ.
2. Calculate $\nabla\Phi$ on Γ.
3. Advance Γ in time using equation (4) .
4. Advance $\Phi(\vec{x_f})$ in time using equation (2).

3.1. Step 1: Solving our BIEs by Collocation

The first stage in a time step is constructing the potential solution satisfying our boundary conditions. We approximate Γ using a triangular mesh comprised of triangles T_j and vertices $\vec{x}_i$. We will refer to a per-triangle enumeration of vertices as $\vec{v}_{j1}, \vec{v}_{j2}, \vec{v}_{j3}, \in T_j$. A collocation scheme simply requires that the integral equations are satisfied at discrete locations: we choose the mesh vertices. This gives the following semi-discrete system:

$$\Phi(\vec{x}_{if}) = \int_\Gamma \sigma(\vec{y}(k)) G(\vec{x}_{if}, \vec{y}(k)) dk, \tag{8}$$

$$\frac{\partial}{\partial n}\Phi(\vec{x}_{is}) = \frac{\sigma(\vec{x}_{is})}{2} + \frac{\partial}{\partial n}\int_\Gamma \sigma(\vec{y}(k)) G(\vec{x}_{is}, \vec{y}(k)) dk. \tag{9}$$

To fully discretize the system requires numerical quadratures for the integrals. For equation (9), the theoretical properties of a second kind integral equation means we can expect numerical stability and well-conditioning with standard quadrature that smooths the singularity.

In contrast, the first kind integral equation (8) poses immediate challenges to effective numerical quadrature. The kernel is integrable with smoothness assumptions on the geometry and σ, but unfortunately any discrete approximation will have to deal with the singularity of the kernel. In addition, theoretically the integral operator is compact and therefore inverting it is ill-conditioned. However, so long as appropriate resolution, regularization or filtering is used, we can still attain meaningful results, following in the footsteps of prior numerical work [XXLY01, GGD01, FD06]. We will attempt to provide intuition for the issue.

It is readily apparent that a kernel whose action smooths the argument σ will result in a first kind equation for which the inverse is ill-posed, since very distinct rough inputs could be smoothed to virtually identical smooth outputs. Thus, one expects that treating the singularity by a smooth approximation will result in an extremely ill-conditioned, if not unsolvable, linear equation, producing arbitrary high-frequency noise even if the low-frequency smooth components are well-determined. Numerically solving such a system, one would expect to see increasing high-frequency noise as the resolution increases as explained by Groetsch [Gro07]. However, in regards to solving the weakly singular integral equation of the first kind such as ours, Atkinson [Atk97] states that they can be mathematically well-behaved and gives examples of numerical treatment that result in a stable numerical system with accurate solution to the two-dimensional analog of our equation.

It is not apparent how Xue et al. handle integrating the singularity in their method, though they mention seeing fine scale noise that they accordingly filter; Fochesato et al. simply state that they use "recursive subdivision". We accurately integrate the singularity similar to Grilli et al. by first applying an integral transformation to remove the singularity, then using Gaussian quadrature. This process is commonly known as Duffy quadrature [Duf82]. In our case, we found it resulted in a well-behaved linear system that gave results consistent with our low-order geometry approximation. In contrast, neglecting the singularity and using standard low order quadrature for the first kind equation resulted in our linear solver failing to converge.

3.1.1. Numerical Quadrature

Our first step in implementing our quadrature is to approximate σ by discrete values σ_i at each mesh vertex, linearly interpolated across triangular faces.

We calculate the equations (8) and (9) at a mesh vertex $\vec{x}_i$ by integrating over all triangles. For those triangles which don't include $\vec{x}_i$ we use a simple 3-point quadrature rule,

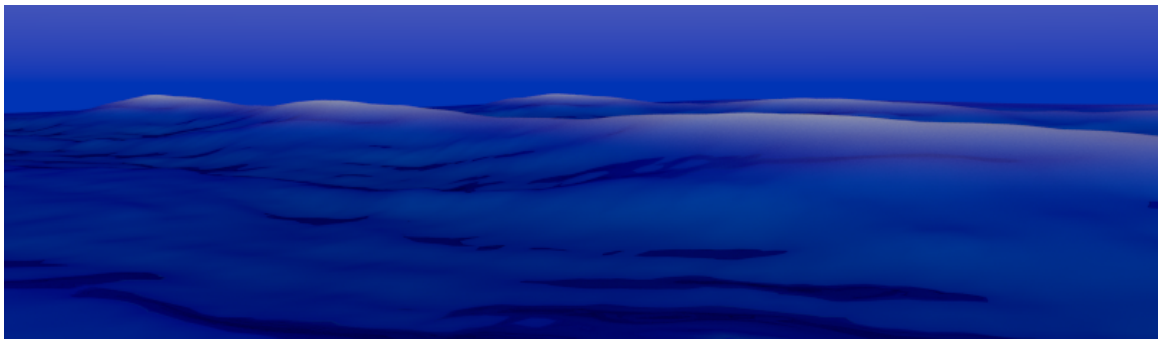

Figure 3: *Ocean Waves initialized with Phillips spectrum [T*01].*

$$\int_{T_j} \sigma(\vec{y}(s))G(\vec{x}_i,\vec{y}(s))ds \approx \frac{1}{3}\left[\sum_{k=1,2,3} \frac{\sigma(v_{jk})}{4\pi|\vec{x}_i-\vec{v}_{jk}|})\right] \mathrm{A}(T_j) \quad x_i \notin T_j, \tag{10}$$

and

$$\int_{T_j} \sigma(\vec{y}(k))\frac{\partial}{\partial n}G(\vec{x}_i,\vec{y}(k))dk \approx \frac{1}{3}\left[\sum_{k=1,2,3} -\sigma(v_{jk})\frac{\vec{n}_{is}\cdot(\vec{x}_{is}-\vec{v}_{jk})}{4\pi|\vec{x}_{is}-\vec{v}_{jk}|^3})\right] \mathrm{A}(T_j) \quad x_i \notin T_j, \tag{11}$$

where $A(T_j)$ is the area of triangle j. When vertex $\vec{x}_i \in T_j$, and we are calculating equation (9) we adapt the 3-point quadrature from equation (11) by setting σ_i to zero at $\vec{x}_i = \vec{v}_{ij}$, giving an $O(h^2)$ approximation for characteristic edge length h. This low order approximation is acceptable since the diagonal elements of the resulting matrix are dominated in the limit by the jump condition in (9). When vertex $\vec{x}_i \in T_j$ and equation (8) is being calculated, we instead use Duffy quadrature, described below, to handle the singularity.

3.1.2. Duffy Quadrature

Duffy quadrature uses a simple variable transformation to completely remove the singularity from the integrand. For the sake of simplicity, we describe Duffy quadrature on a reference triangle T_r in the x-y plane, $v_{r1} = (0,0,0)$, $v_{r2} = (1,1,0)$, and $v_{r3} = (0,0,1)$, where the singularity is at v_{r1}. All other triangles can be affinely transformed to this reference triangle. On the reference triangle we can write the right hand side of (6) as the 2D integral

$$\frac{1}{4\pi}\int_0^1 \int_0^x \frac{\sigma(x,y)}{\sqrt{x^2+y^2}}\,dx\,dy. \tag{12}$$

With the transformation, $u = y/x$, we arrive at

$$\frac{1}{4\pi}\int_0^1 \int_0^1 \frac{\sigma(x,xu)}{\sqrt{x^2+(xu)^2}}\,x\,dx\,du = \frac{1}{4\pi}\int_0^1 \int_0^1 \frac{\sigma(x,xu)}{\sqrt{1+(u)^2}}\,dx\,du. \tag{13}$$

The singularity in equation (12) no longer exists in (13). We approximate (13) with a nine-point tensor Gaussian quadrature on the square, using the linearly-interpolated σ. The linear transformation into the reference triangle of course needs to be taken into account in general.

3.1.3. Linear Solver

The result of applying quadrature to equations (8) and (9) is a dense linear system. Actually constructing the dense matrix for a direct solve would be unacceptably expensive. We instead look to iterative solvers that don't require the whole matrix but merely compute several matrix-vector products, choosing BiCGSTAB [vdV92] in our implementation. Moreover, we can further accelerate matrix-vector multiplication with parallelism and the Fast Multipole Method, described in section 4. For smooth free-surface-only geometry we found BiCGSTAB solved the systems to 10^{-5} of the initial residual norm in less than 30 iterations. With solid geometry included, it typically took 50–60 iterations. However, when the mesh becomes overly distorted with sliver triangles, the iteration count can rise past 100, and if the simulation becomes unstable and the geometry pathological, the solver can fail to converge. We thus employ the El Topo library to maintain a well-formed mesh by restricting maximum and minimum edge lengths, and further stabilize the numerical method by adding some viscosity described below.

3.2. Step 2: Computing $\nabla\Phi$

The normal derivative at the surface is given by equation (7) and can be calculated by applying the same process describe for computing (9). The derivatives in the tangential

directions $\vec{t}$ we compute by applying the gradient directly to equation (5). There is no jump in the tangential derivatives as they approach the boundary, yielding

$$\vec{t}_i \cdot \nabla\Phi(\vec{x}_{i\Gamma}) = \vec{t} \cdot \nabla \int_\Gamma \sigma(\vec{y}(k)) G(\vec{x}_{i\Gamma}, \vec{y}(k))\, dk. \tag{14}$$

We compute this in the same low-order fashion as before,

$$\vec{t}_i \cdot \nabla \int_{T_j} \sigma(\vec{y}(k)) G(\vec{x}_i, \vec{y}(k))\, dk \approx \frac{1}{3}\left[\sum_{k=1,2,3} -\sigma(v_{jk}) \frac{\vec{t}_i \cdot (\vec{x}_{i\Gamma} - \vec{v}_{jk})}{4\pi|\vec{x}_{i\Gamma} - \vec{v}_{jk}|^3}\right] \mathrm{A}(T_j) \quad x_i \notin T_j. \tag{15}$$

Again we neglect the singular components when $\vec{x}_i \in T_j$ by setting σ_i to zero there. We compute the tangential derivative for two perpendicular tangential components then combine them with the already computed normal component to construct the gradient of the potential on the boundary. The engineering literature computes the tangential gradients by making a local high-order approximation to the surface from which they can approximate the tangential derivatives using the necessary geometric information and values of Φ. See especially the method by Grilli et al. [GGD01]. Our method is a far simpler way of constructing the tangential derivative and more robust for unstructured triangle meshes where local high-order approximation to the surface would be a challenge.

3.3. Steps 3 and 4: Time Integration

We update the surface and potential at time $t = \Delta t \cdot (n+1)$ using symplectic Euler:

$$\vec{x}_i^{n+1} = \vec{x}_i^n + \Delta t \nabla \Phi_i^n \tag{16}$$

$$\Phi_i^{n+1} = \Phi_i^n + \Delta t \left[p(\vec{x}_i^{n+1}) - gz(\vec{x}_i^{n+1})\right]. \tag{17}$$

We further stabilize the system by including (in essence) some artificial viscosity, adaptively smoothing the mesh in the tangential plane. This is done by computing a weighted average of each vertex's position with its m neighbors,

$$\vec{\bar{x}}_i = \alpha \vec{x}_i + \beta \sum_{k=1}^{m} \vec{x}_k, \tag{18}$$

$$\beta = (1-\alpha)/m, \tag{19}$$

and then moving the vertex to the projection of the weighted average on the tangent plane. We found that smoothing in the tangential plane resulted in well formed triangles without noticeable energy loss due to the altitude erosion that one would see with regular smoothing. On the solid surface the boundary values of Φ is not propagated in the model and the fluid geometry can be changed per time-step without affecting the solution. Here we tangentially smooth quite strongly with values of $\alpha = 0.05$. On the free surface, we generally smooth with α values around .97 for a time-step of $dt = .01$, computing four simulation time-steps per frame. At the edge of our simulation, we smooth Φ and the mesh in all planes to avoid problems when large waves hit the mesh edge. At the edge points we smooth only in the y-coordinate, but still allow motion due to the potential gradient.

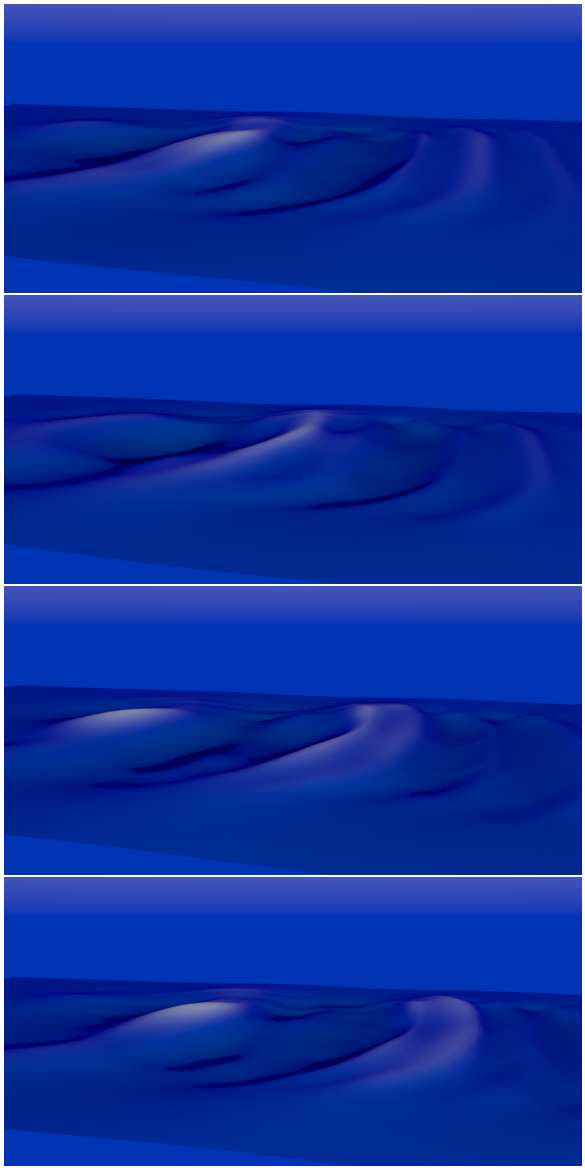

Figure 4: *Sequence of images showing waves with longer wavelength overtaking those with smaller, a characteristic of deep water dispersion.*

3.4. Solid Boundaries and Mesh Adaptivity

To determine whether a fluid point is a solid or free boundary, we check auxiliary meshes denoting solid objects for collision with the fluid mesh. Potential collisions are handled by the El Topo mesh library. A fluid point that is in an adjustable proximity to a solid becomes considered part of the solid boundary and is moved to a pre-defined distance from the solid. Solid normal velocities are then applied to this vertex during time-stepping and it is considered to have the requisite Neumann boundary condition in the numerical solver.

For deep ocean waves, we found that the natural tangential motion of the wave dynamics caused the compression of the surface mesh near areas of high curvature such as the peaks of the waves. This gives an added efficiency to the method, imparting an effective resolution many times higher then the original mesh size. While this adaptivity can sometimes be reversed due to unnatural pressure forcing, it is consistent for areas where the natural wave motion is allowed to occur.

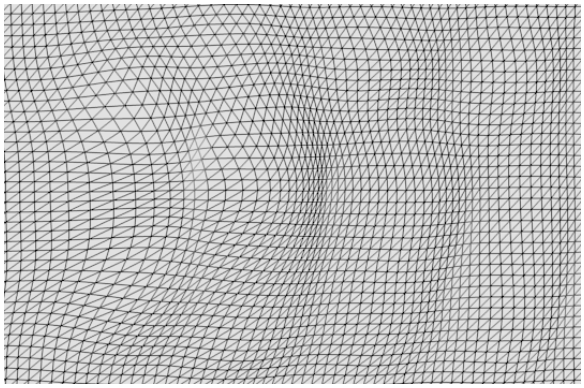

Figure 5: *Top down view of the computational mesh showing natural adaptivity induced by Lagrangian motion.*

4. FMM and GPU acceleration

Boundary integral equations reduce the geometry and thus memory requirements of wave simulation, from a 3D grid to just the 2D surface mesh, and exhibit nice convergence properties with using iterative solvers. However, the $O(N^2)$ complexity of multiplying with the (conceptually) dense matrix means we've essentially moved complexity from space into time. Fortunately, we can reduce the high compute cost of solving boundary integral methods. We employ two different approaches, using the FMM and also GPU acceleration to more efficiently compute the wave simulation. In both cases, instead of computing integrals by summing per triangle as described, we first compute all the quadrature weights accumulated by a vertex from its neighboring triangles, and then

Dims	Verts	FMM time	GPU time
80x80	6400	6.6	6.0E-2
160x160	25600	30	6.3E-1
320x320	102400	96	9.0
640x640	409600	3.5E2	1.3E2

Table 1: *seconds per BiCGSTAB iteration using FMM or GPU acceleration for mesh dimensions (Dims) and total number of vertices (Verts)*

compute the integrals by summing over each vertex. Computing the quadratures in this fashion becomes an N-body problem involving the fundamental solution for Laplace's equation and its gradient. This can be done in $O(N)$ time using the FMM. In addition, the N-body problem is embarrassingly parallel and is straightforward to implement in OpenCL to exploit the massive parallelism of modern GPUs.

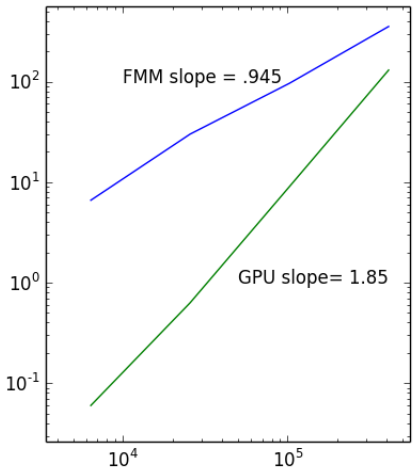

Figure 6: *Logarithmic plot of table 1 showing the number of vertices versus time per iteration asymptotics of the FMM and the GPU.*

The FMM we used ran on a single CPU thread on a 3.2Ghz Intel i5-3470 at 3-4 digits of accuracy; though it showed linear-time asymptotics, as seen in table 1 and Figure 6, it was still slower on our examples then the direct GPU-accelerated version using an AMD Radeon 7970. We are currently working on a GPU version of the FMM to combine these two acceleration techniques for added computational efficiency.

An additional advantage of our BIE formulation is that it allows us to easily use our FMM, implemented according to that described in [CGR99]. The formulations in the engineering literature require dipole charges rather then single

point sources, which necessitates non-trivial modifications to the numerical machinery of the usual FMM.

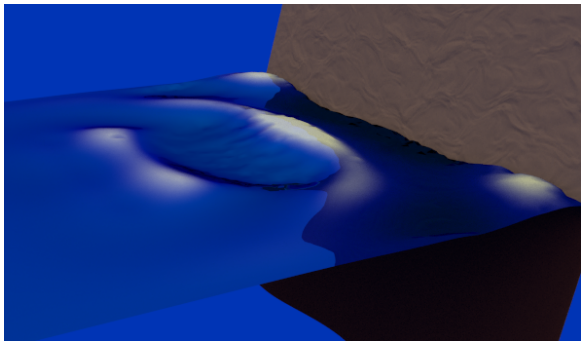

Figure 7: *Waves incident upon a near-vertical wall.*

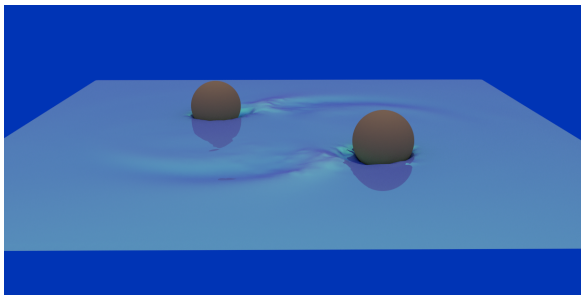

Figure 8: *Moving balls with wake.*

5. Results

Our results demonstrate deep ocean wave phenomena. Figure 1 shows the steep peaks and shallow troughs expected for ocean waves. Figure 3 shows a large scale simulation of waves initialized with a Phillips spectrum [T*01]. The dispersion relationship for deep water waves imparts higher wave speed to waves of longer wavelength. Figure 4 shows this happening in our simulation. While this visually apparent dispersion relationship is inherent in FFT simulations and evident in ours, it is not captured by shallow water simulations. Figure 9 shows a comparison with a height-field waves simulation using FFT's. Figure 7 shows waves crashing on a near vertical wall, demonstrating solid boundary interactions. Figure 8 shows two balls moving on the water surface, demonstrating moving solid boundary interactions. Figure 5 demonstrates the natural adaptivity of our method, allowing a regular grid to distort naturally, compressing the grid at the sharp peaks of the waves and decompressing it in the smoother troughs.

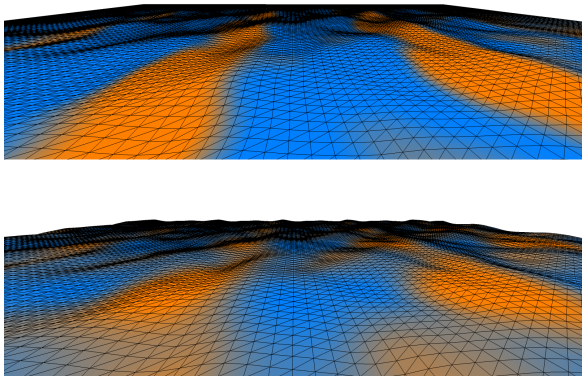

Figure 9: *Comparison of our waves (top) with a Tessendorf/FFT height-field wave simulation (bottom) using the same initial conditions. Coloring represents the potential values on the surface .*

6. Future Work

Our BIE method is a novel method for deep ocean wave animation, and thus there are ample avenues for future work:

- The greatest immediate barrier to effectiveness of the method is its relatively high compute cost. As we've shown, the FMM and GPU computing can ameliorate that. Combining the two could bring us closer to large-scale, low-memory ocean simulations with dynamic solid boundaries.
- The method is dependent on good mesh-tracking software. While El Topo was well suited to the current method, phenomena such as overturning waves will need meshing algorithms that handle topology changes robustly and immediately, without leaving thin interfaces of air between fluids as El Topo is wont to do.
- Stable ways to incorporate the non-linear term would expand the use cases of the method. While the non-linear term is not necessary for general ocean simulation, specific phenomena such as overturning waves and wave-breaking would require implementing this term and provide interesting problems in their own rights.
- Tight integration with Tessendorf waves to continue ocean simulations out to the horizon would be a straightforward next step. It is simple to derive the surface potential values from a FFT based wave simulation; one could easily use this information to drive the boundaries of our method. This would allow seamless interfaces between the more dynamic and interactive BIE method and a periodically extended FFT-based procedural ocean.
- Non-reflecting conditions on the edge of the domain where the free surface continues.
- Two-way coupling with buoyant objects would allow

more dynamic ocean scenes. This would require developing a buoyancy model compatible with potential flow.

Acknowledgements

We would like to acknowledge Mitacs and Microsoft Studios for their generous funding which partially supported this project.

References

[ATBG08] ANGST R., THUEREY N., BOTSCH M., GROSS M.: Robust and efficient wave simulations on deforming meshes. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 1895–1900. 2

[Atk97] ATKINSON K. E.: *The numerical solution of integral equations of the second kind*. No. 4. Cambridge university press, 1997. 4

[BB09] BROCHU T., BRIDSON R.: Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4 (2009), 2472–2493. URL: http://link.aip.org/link/?SCE/31/2472/1, doi:10.1137/080737617. 2

[BKB12] BROCHU T., KEELER T., BRIDSON R.: Linear-time smoke animation with vortex sheet meshes. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2012), The Eurographics Association, pp. 87–95. 2

[CGR99] CHENG H., GREENGARD L., ROKHLIN V.: A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics 155*, 2 (1999), 468–498. 7

[CM10] CHENTANEZ N., MÜLLER M.: Real-time simulation of large bodies of water with small scale details. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation* (2010), Eurographics Association, pp. 197–206. 2

[CM11] CHENTANEZ N., MÜLLER M.: Real-time eulerian water simulation using a restricted tall cell grid. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 82. 2

[Duf82] DUFFY M.: Quadrature over a pyramid or cube of integrands with a singularity at a vertex. *SIAM journal on Numerical Analysis 19*, 6 (1982), 1260–1262. 4

[FD06] FOCHESATO C., DIAS F.: A fast method for nonlinear three-dimensional free-surface waves. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science 462*, 2073 (2006), 2715–2735. 2, 4

[FR86] FOURNIER A., REEVES W. T.: A simple model of ocean waves. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 75–84. URL: http://doi.acm.org/10.1145/15922.15894, doi:10.1145/15922.15894. 2

[GGD01] GRILLI S. T., GUYENNE P., DIAS F.: A fully non-linear model for three-dimensional overturning waves over an arbitrary bottom. *International Journal for Numerical Methods in Fluids 35*, 7 (2001), 829–867. 2, 4, 6

[GNS*12] GOLAS A., NARAIN R., SEWALL J., KRAJCEVSKI P., DUBEY P., LIN M.: Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 148. 2

[GR87] GREENGARD L., ROKHLIN V.: A fast algorithm for particle simulations. *Journal of Computational Physics 73* (December 1987), 325–348. 2

[Gro07] GROETSCH C.: Integral equations of the first kind, inverse problems and regularization: a crash course. In *Journal of Physics: Conference Series* (2007), vol. 73, IOP Publishing, p. 012001. 4

[IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 805–811. 2

[JP99] JAMES D. L., PAI D. K.: Artdefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 65–72. 2

[LvdP02] LAYTON A. T., VAN DE PANNE M.: A numerically efficient and stable algorithm for animating water waves. *The Visual Computer 18*, 1 (2002), 41–53. 2

[LXY01] LIU Y., XUE M., YUE D.: Computations of fully nonlinear three-dimensional wave-wave and wave-body interactions. part 2. nonlinear waves and forces on a body. *Journal of Fluid Mechanics 438* (2001), 41–66. 2

[NB11] NIELSEN M. B., BRIDSON R.: Guide shapes for high resolution naturalistic liquid simulation. *ACM Trans. Graph. 30*, 4 (July 2011), 83:1–83:8. URL: http://doi.acm.org/10.1145/2010324.1964978, doi:10.1145/2010324.1964978. 2

[Pea86] PEACHEY D. R.: Modeling waves and surf. In *ACM Siggraph Computer Graphics* (1986), vol. 20, ACM, pp. 65–74. 2

[PK05] PARK S. I., KIM M. J.: Vortex fluid for gaseous phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 261–270. URL: http://doi.acm.org/10.1145/1073368.1073406, doi:http://doi.acm.org/10.1145/1073368.1073406. 2

[T*01] TESSENDORF J., ET AL.: Simulating ocean water. *Simulating Nature: Realistic and Interactive Techniques. SIGGRAPH* (2001). 1, 2, 5, 8

[vdV92] VAN DER VORST H. A.: Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput. 13*, 2 (Mar. 1992), 631–644. URL: http://dx.doi.org/10.1137/0913035, doi:10.1137/0913035. 5

[WP10] WEISSMANN S., PINKALL U.: Filament-based smoke with vortex shedding and variational reconnection. *ACM Transactions on Graphics (Proc. SIGGRAPH) 29* (July 2010), 115:1–115:12. URL: http://doi.acm.org/10.1145/1778765.1778852, doi:http://doi.acm.org/10.1145/1778765.1778852. 2

[XXLY01] XUE M., XU H., LIU Y., YUE D.: Computations of fully nonlinear three-dimensional wave-wave and wave-body interactions. part 1. dynamics of steep three-dimensional waves. *Journal of Fluid Mechanics 438* (2001), 11–39. 2, 4

[YHK07] YUKSEL C., HOUSE D. H., KEYSER J.: Wave particles. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 99. 2