Surface Turbulence for Particle-Based Liquid Simulations

Olivier Mercier[1] Cynthia Beauchemin[1] Nils Thuerey[2] Theodore Kim[3] Derek Nowrouzezahrai[1]
[1] Université de Montréal [2] Technische Universität München [3] University of California, Santa Barbara

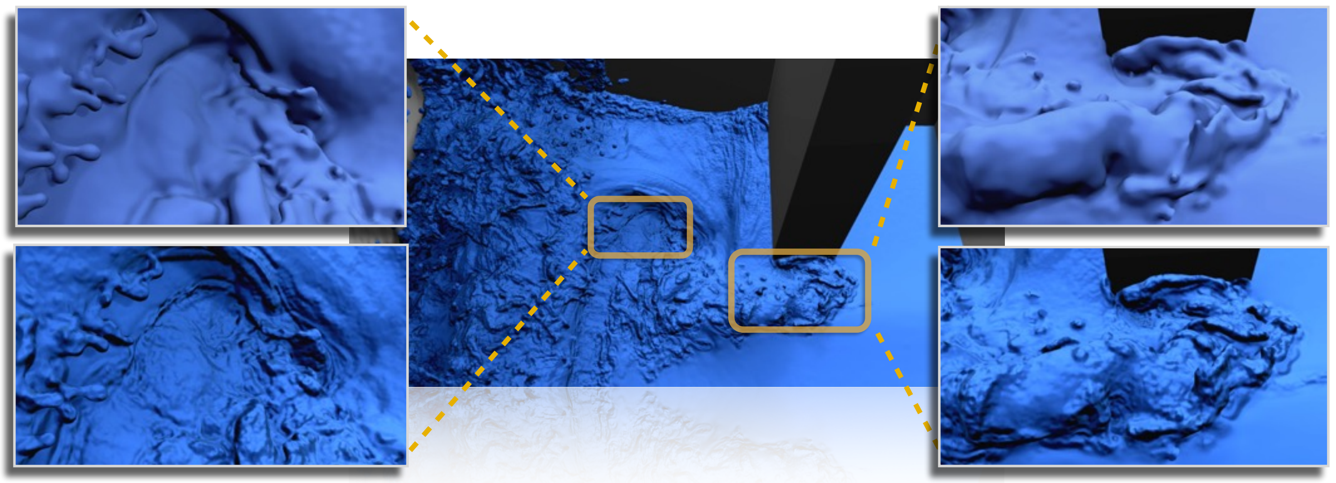

Figure 1: *We apply our turbulence model to a high-resolution FLIP simulation ($> 12\times10^6$ particles). Zoom-ins compare the unmodified input surface (top) to our output (bottom). Even at high resolutions, the input simulation fails to resolve small scale details, which our method is capable of adding. In this extreme example, our entire post-process adds an overhead of roughly a third of the full simulation time.*

Abstract

We present a method to increase the apparent resolution of particle-based liquid simulations. Our method first outputs a dense, temporally coherent, regularized point set from a coarse particle-based liquid simulation. We then apply a surface-only Lagrangian wave simulation to this high-resolution point set. We develop novel methods for seeding and simulating waves over surface points, and use them to generate high-resolution details. We avoid error-prone surface mesh processing, and robustly propagate waves without the need for explicit connectivity information. Our seeding strategy combines a robust curvature evaluation with multiple bands of seeding oscillators, injects waves with arbitrarily fine-scale structures, and properly handles obstacle boundaries. We generate detailed fluid surfaces from coarse simulations as an independent post-process that can be applied to most particle-based fluid solvers.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: Lagrangian fluid simulation, wave turbulence

1 Introduction

Simulating the behavior of fluids is a long standing problem that often requires visual details resolved to a very fine resolution. Simulating smoke and liquids are the two most common cases, and specialized approaches have been developed for each.

For *smoke* animation, Eulerian approaches are common. Here, performance scales with the underlying grid's resolution and resolving details at fine scales quickly becomes prohibitive. *Fluid up-res* methods address this problem by applying fine-scale turbulence models atop coarser simulations, generating detailed results without explicitly simulating at a fine resolution [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008]. These methods are practical for art-direction since they guarantee that the coarse behavior will not change when fine details are added.

In contrast, detailed *liquid* simulations are still performed at full resolution, regardless of whether a grid- or particle-based approach is used, with high-quality particle-based simulators, such as Fluid Implicit Particle (FLIP) [Zhu and Bridson 2005; Autodesk 2014] or Smoothed Particle Hydrodynamics (SPH) [Müller et al. 2003; Next Limit Technologies 2014], having seen rapid, recent adoption.

We address the discrepancy with an *"up-res"* technique for particle-based liquid simulations. While an Eulerian *closest point turbulence* (CPT) method was recently developed for level set-based liquids [Kim et al. 2013], it can only be applied to particle-based data by converting the data to an Eulerian grid, discarding many of the simulation's rich details. We instead add turbulent details *directly* to particles, solving the wave equation in a Lagrangian setting. We first convert a set of input particles from a coarse liquid simulation into a high-quality, high-density surface point set. We then perform a wave simulation that adds high-frequency features to the liquid surface, in the form of bump or displacement maps. We use standard surfacing to obtain a detailed, high-resolution liquid surface.

To our knowledge, ours is the first comprehensive up-res technique for particle-based simulations, making the following contributions:

- robust, temporally coherent, meshless surface generation, that yields smooth, simulation-ready surfaces,
- smooth constraints to ensure that our surface remains spatially and temporally faithful to the underlying particle set,
- a robust, curvature-based method for initiating surface waves,
- a novel discrete Laplace operator that is provably well-suited for meshless point representations, in addition to
- an efficient simulation strategy that synthesizes details across scales onto our high-density surface.

Our method is agnostic to the source of particle data, and can be applied to FLIP, SPH, and position-based works [Macklin et al. 2014].

2 Previous Work and Overview

Fluid simulation is a well-established area so, in addition to seminal works [Foster and Metaxas 1996; Stam 1999; Foster and Fedkiw 2001; Müller et al. 2003], we refer readers to Bridson's book [2008] and Ihmsen et al.'s STAR [2014] for comprehensive surveys. Here, we focus primarily on areas most related to our work.

Fluid Up-res. Thuerey et al. [2013] surveys recent fluid up-res methods which efficiently increase the apparent spatial resolution of a coarse simulation without altering the low-frequency behavior. As noted in Section 1, these methods have been most effective in smoke simulations [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008; Nielsen et al. 2009; Huang et al. 2011]. Several works have also addressed the related problem of synthesizing frequency-controlled textures on moving surfaces [Yu et al. 2009].

Earlier attempts to apply up-res algorithms to liquids had limited success, both in Eulerian [Narain et al. 2008] and Lagrangian [Yuan et al. 2012; Shao et al. 2014] formulations, since they focus on increasing the resolution of the velocity field. As noted by Kim et al. [2013], the turbulence on a free surface is only loosely coupled to the fluid velocity field. Lab experiments show that surface waves tend to propagate much faster than the velocities of the underlying fluid would suggest. We thus choose to evolve a high-resolution simulation over the liquid surface to obtain more convincing results.

While CPT [Kim et al. 2013] works well for Eulerian liquids, no comprehensive Lagrangian up-res method exists. This is unfortunate, because the ad-hoc methods developed in industry [Budsberg et al. 2013] show that there is substantial interest in such techniques.

Several works have explored how to guide liquids to meet artistic goals [Shi and Yu 2005; Pan et al. 2013]. Nielsen and Bridson [2011] use a low-frequency "guide shape", extracted from a coarse FLIP simulation, as the boundary condition for a thin, secondary, high-resolution simulation applied near the liquid boundary. Our work differs from this approach in two ways: first, we preserve the entire frequency content of the coarse simulation, including important quantities such as the silhouettes. Second, we add entirely new dynamics to the surface using a wave simulation. As such, our algorithm can complement such "guide shapes" approaches, especially since it is applied as a standalone post-process.

Surface Tracking. The importance of surface-only simulations has become increasingly clear in recent works on explicit surface tracking [Thuerey et al. 2010], and methods that use them [Yu et al. 2012]. Wojtan et al. [2011] survey these recent developments. Instead of explicitly modeling the fluid surface and carefully incorporating it into the core simulation, we propose a post-process that can be applied directly to any coarse particle simulation, remaining *fully decoupled* from the simulator that generated the data. Maintaining a simulation mesh is cumbersome, often requiring external geometry processing tools. Our meshless method is self-contained, simplifying implementation. While we create a mesh for rendering, meshing artifacts do not degrade the stability of the simulation.

Inspired by optimization work for liquid surfacing [Williams 2008; Bhatacharya et al. 2011], we constrain our final surface to lie in a band around the input surface. We extend ideas from Eulerian level sets and meshes to particle-based surfaces. While these works focus on generating geometry for rendering, we improve their smoothness and temporal regularity to make them suitable for simulation.

Wave Simulation. Many works consider wave simulations. From the linear wave equation [Kass and Miller 1990], and related shallow-water models [Wang et al. 2007], to bi-Laplacian variants [Yu et al. 2012] and the *iWave* [Tessendorf 2008]. We use the linear wave equation and rely on dispersion from stretching induced by the underlying advection. As with all these previous works, we manually set an average propagation speed for our surface waves.

On the other hand, *wave particles* represent surface waves [Yuksel et al. 2007; Cords 2008] with a dense set of advected points. These methods have difficulty with the complex, topologically-varying surfaces that we treat, requiring many more wave particles to represent the details we achieve with our approach.

Point-Based Simulation. Our work is related to point-based techniques, but unlike previous works that deal with static point sets [Zwicker et al. 2002; Alexa et al. 2003] or dynamic surfaces for rendering [Guennebaud et al. 2008], our input particles represent a volume where every particle corresponds to a quantity of liquid. Point-based rendering treats each sample as a (possibly noisy) surface point, and previous point-based simulations [Macdonald et al. 2013; Jeong and Kim 2013] operate on static point sets, and are thus not appropriate for dynamic particle sets from liquid simulations.

Overview. Figure 2 is a visual overview of our method. Given an input sequence of particles representing a liquid volume (*coarse particles*, dashed circles), we first construct a dense point set along the liquid interface (*fine surface points*, solid circles; Sections 3.2 & 3.3). We smooth and regularize these points to support point-based simulation (Section 3.4). Using per-point normals and displacement values we call *wave values* (green lines; Section 4.2), we perform a high-resolution wave simulation (green curve; Section 4.3) over the surface. We output the final detailed surface as a bump or height map over the high-density point set, or as a displaced point set. These points can be splatted directly or tessellated for rendering, which is easy given our regular surface point distribution.

3 Surface Construction and Maintenance

We construct a dense point set that represents the liquid's surface, and maintain a level of smoothness and regularity necessary for point-based wave simulation. We describe our novel smooth band constraint that controls the surface's behavior and ensures coherence between the surface points and underlying simulation. Unlike level set or mesh-based surface tracking [Osher and Fedkiw 2003; Wojtan et al. 2011], our fluid surface is represented exclusively by oriented points i, each with a position $\mathbf{x}_i$ and normal $\mathbf{n}_i$.

3.1 Neighborhood Relationships

To ensure that our surface points behave as a unified manifold, we draw upon work in meshless simulation (e.g., [Ihmsen et al. 2014]), taking advantage of the neighborhood relationships between point pairs. Our high-resolution surface points $\mathbf{x}_i$, and coarse input particles $\mathbf{X}_i$, will affect each other across spatial scales. Specifically:

- a coarse-scale length λ_c, obtained from the coarse particle simulator (e.g., the grid cell size in a FLIP solver), and

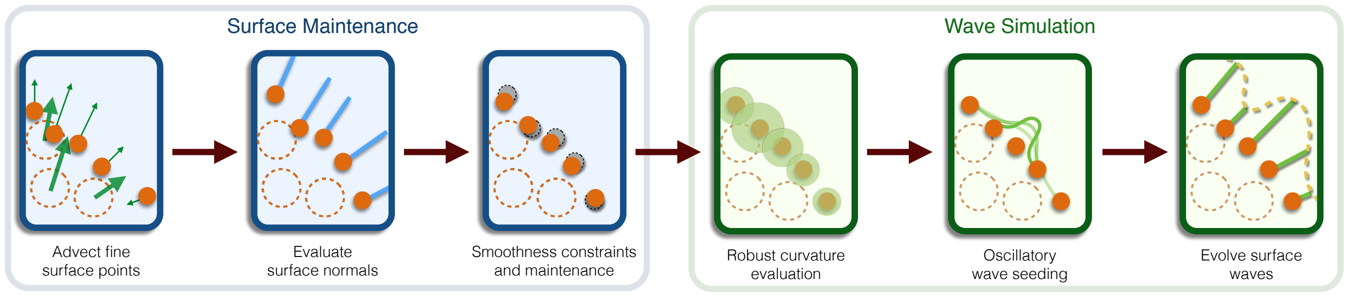

Figure 2: *An overview of the different steps of our algorithm, performed for each frame of coarse input data. The fine surface points (solid circles) are evolved on the surface of the input coarse particles (dashed circles). The overview in Section 2 details each stage of this diagram.*

- a fine-scale length λ_f, a user parameter controlling the separation between points of the detail-enhanced surface.

We use λ_c for operations related to the underlying fluid, such as surface advection (Section 3.2) and curvature evaluation (Section 4.1), and λ_f for intrinsic surface operations, including point distribution regularization (Section 3.4) and wave evolution (Section 4.3). We use isotropic kernels to weight the effect of particles on their neighbors. Unless specified otherwise, we use a simple triangular kernel:

$$K_i^{\delta}(\mathbf{x}_j) = 1 - ||\mathbf{x}_i - \mathbf{x}_j|| / \delta, \text{ if } ||\mathbf{x}_i - \mathbf{x}_j|| < \delta \text{ ; 0, otherwise} , \quad (1)$$

where δ is the local neighborhood radius. We normalize the weighting kernel according to the local density around a point j,

$$\rho_j^{\delta} = \sum_{k \in \mathcal{F}} K_j^{\delta}(\mathbf{x}_k) , \quad (2)$$

where $\mathcal{F}$ denotes the set of all surface points. This local normalization eliminates any bias introduced by potential non-uniformities across local point densities, and so the local weight is $w_i^{\delta}(\mathbf{x}_j) = K_i^{\delta}(\mathbf{x}_j)/\rho_j^{\delta}$. This density normalization is especially important at the finest scale, where point distribution variance is highest.

We also normalize the weighting so the contributions sum to unity, so our final weight of point j for point i is

$$W_i^{\delta}(\mathbf{x}_j) = w_i^{\delta}(\mathbf{x}_j) \Big/ \sum_{k \in \mathcal{F}} w_i^{\delta}(\mathbf{x}_k) . \quad (3)$$

In practice, we only sum over particles in finite neighborhoods due to the local support of K. The weights also apply naturally to coarse particles by exchanging $\mathcal{F}$ for $\mathcal{C}$, the set of all coarse particles.

3.2 Surface Initialization and Advection

We create the initial set of fine-scale surface points in two steps. We first create an initial point set by centering spheres of radius λ_c around each coarse particle, sampling points uniformly on these spheres, and only retaining points that do not fall inside any other sphere's volume. This process results in a very rough, non-smooth, fine-scale surface point distribution that we regularize in a second step (see Section 3.4). For any subsequent frame n, fine surface points $\mathbf{x}_i^n$ are obtained by simply advecting points $\mathbf{x}_i^{n-1}$ from the previous frame. The updated surface point position is a weighted sum of the displacements of neighboring coarse particles $\mathbf{X}$,

$$\mathbf{x}_i^n \leftarrow \mathbf{x}_i^{n-1} + \sum_{k \in \mathcal{C}} W_i^{2\lambda_c}(\mathbf{X}_k) \left(\mathbf{X}_k^n - \mathbf{X}_k^{n-1}\right). \quad (4)$$

We can modify the neighborhood size used for advection according to how closely we want fine-scale surface points to track coarse particles; in practice, a neighborhood radius of $2\lambda_c$ yields smoothly advected fine-scale surfaces. We use this value for all of our results.

After advection, fine-scale points may no longer be located in the vicinity of the coarse particles, so the fine-scale surface behavior may deviate from the dynamics of the underlying coarse simulation. Maintaining a correspondence between the input (coarse) and output (fine) fluid behavior is essential for predictable artistic control, so we next devise a surface constraint that imposes this guarantee.

3.3 Surface Constraints

A fundamental component of our approach is the generation of a smooth, temporally coherent, high-resolution output surface that does not need to explicitly track manifold connectivity. To accomplish this, we devise an implicit representation that constrains the position of the fine-scale surface points.

Our method is motivated by Williams [2008]: First, two concentric spheres of radius r and R are centered at each coarse particle, where R is the larger of the two. During surface evolution, the fine-scale surface points are constrained to remain in the volume between the union of all outer spheres and the union of all inner spheres. As depicted in Figure 3, this volume region forms "bands" around the coarse particles, delimiting the region where the advected fine scale surface is allowed to exist. This constrains surface points to regions not too far from, nor too close to, the existing coarse particles.

The r and R parameters affect the final output appearance. Small values create surfaces that more closely resemble the coarse simulation, but increase bumpiness. Large values can create surfaces that deviate significantly from the coarse simulation and exhibit over-smoothing. While they can be manually adjusted, we found that $R = \lambda_c$ results in a reasonably smooth surface relative to the underlying simulation and $r = R/2$ leaves sufficient space for fine particles to evolve without closely approaching the coarse particles.

Williams uses the spheres to constrain a thin-plate energy optimization to represent the surface; it is unclear how this can apply to meshless settings without costly intermediate meshing. Moreover, projecting onto the surface formed by the union of spheres is difficult due to discontinuities in the first derivatives, and an orthogonal projection creates a discontinuous surfaces unsuitable for wave simulation (Section 4.3).

To solve this problem, we instead project onto an implicit metaball-like formulation [Blinn 1982] that leads to smooth projection constraints better suited to the volumetric region between the inner- and outer-sphere unions. The efficacy of this strategy is shown in Figure 3. This novel *smooth band constraint* is constructed from an implicit function $\phi(\mathbf{y}) = g(f(\mathbf{y}))$ for an arbitrary point $\mathbf{y}$, where f is a standard metaball function and g is a rescaling function. There are many possible choices for f, but we obtained good results with

$$f(\mathbf{y}) = \sum_{i \in \mathcal{C}} f_i(\mathbf{y})/\psi_i \quad \text{with} \quad f_i(\mathbf{y}) = \exp\left(-a|\mathbf{y} - \mathbf{X}_i|^2\right) , \quad (5)$$

where ψ_i is the metaball density of the i-th coarse particle and a is a falloff parameter discussed below. The metaball density ψ_i differs from the local density ρ_i^{δ} in Equation 2, and is evaluated as

$$\psi_i = \sum_{j \in \mathcal{C}} D(||\mathbf{x}_i - \mathbf{x}_j||) \quad \text{with} \quad D(z) = \exp\left(-2\ (z/\lambda_c)^2\right) . \quad (6)$$

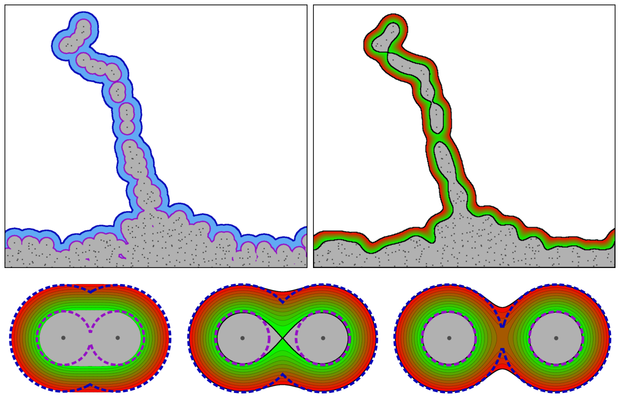

Figure 3: *Williams' constraint circles (top left, bottom dashed circles) and our smooth band constraint (top right, bottom color gradient). Our constaints are smoother and define values in the interior that vary linearly in space, permitting simpler projections.*

Using different kernels for $f_i(\mathbf{y})$ and $D(z)$ provides the flexibility needed to design sufficient constraints. As in Equations 1 to 3, we sum over local surface point neighborhoods in Equations 5 and 6 due to the kernel's fall-off. Unlike before, truncation *does* introduce error, but only a negligible amount at a neighborhood radius of $2R$.

Finally, we impose an almost linear variation in ϕ, from 0 at the inner sphere to 1 at the outer sphere, as illustrated in Figure 3. We use the following scaling function, which is exact for a single particle:

$$g(z) = \left(\sqrt{-\ln(z)/a} - r\right) \Big/ (R - r)\,. \tag{7}$$

We solve for a value of a that, given two coarse particle centers less than $\mu = {}^3\!/_2 \times R$ units apart, will unify the inner sphere union constraint of the two particles as if their contribution resulted from the same component of the fluid surface:

$$a = \ln\left(2/[1 + D(\mu)]\right) \Big/ \left(({}^{\mu}\!/_2)^2 - r^2\right). \tag{8}$$

Figure 3 (bottom middle) is the case with particle centers exactly μ units apart. All our results use this value of a, and its associated μ.

Since our band constraint function ϕ is smooth, we can use its normalized gradient $\mathbf{g}$ to determine a reasonable projection direction when a particle exits the constraint region. Additionally, as the function is approximately linear with respect to the distance from the inner constraint, we can easily compute this projection as

$$\mathbf{x}_i \leftarrow \begin{cases} \mathbf{x}_i - (R - r)\cdot\phi(\mathbf{x}_i)\cdot\mathbf{g}_i & \text{if } \phi(\mathbf{x}_i) < 0 \\ \mathbf{x}_i - (R - r)\cdot(\phi(\mathbf{x}_i) - 1)\cdot\mathbf{g}_i & \text{if } \phi(\mathbf{x}_i) > 1 \\ \mathbf{x}_i & \text{otherwise}\,. \end{cases} \tag{9}$$

This introduces some approximation error, so a surface point may still lie outside the constraint region after projection. However, at this stage, it keeps points sufficiently close to the constraint region.

3.4 Surface Smoothing and Regularization

We now have a high-density set of surface points, but their distribution must be improved (i.e. regularized) before they are suitable for surface wave simulation. This regularization proceeds in four stages: normal computation, normal regularization, tangent regularization, and point insertion and deletion.

Normal Computation: A smooth, artifact-free normal field is essential for the maintenance of our surface structures. We compute the normal at each fine-scale surface point using an averaged least-squares planar fit to the local gradient of ϕ (see Algorithm 1).

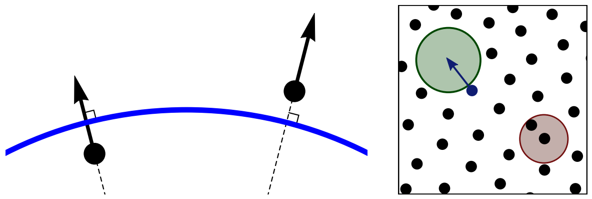

Figure 4: *Left: Surface regularization shifts points along their normal towards circles consistent with the points' positions and normals, smoothing the surface. Right: A surface point added to a low density region (green) and deleted from a high density region (red).*

Regularization Along the Normal: To improve the smoothness of our surface, we displace each surface point along its newly computed normal direction. Given a point i and one of its neighbors j, we consider the plane $\Pi_{i,j}$ spanned by vectors $\mathbf{n}_i$ and $(\mathbf{x}_j - \mathbf{x}_i)$. We find the unique circle in this plane that is equidistant to both points and orthogonal to both points' normals (see Figure 4). The projection of point i onto this circle is averaged over all its neighbors, and the point is then displaced along its averaged projected position.

This averaging is done in a neighborhood of radius λ_c, which pushes the points towards a surface that is consistent with the computed normal field. Since the normals vary smoothly, the resulting surface is also smooth. Denoting $\mathbf{n}_j^\star$ the normalized projection of $\mathbf{n}_j$ onto $\Pi_{i,j}$, the displacement of point i onto the circle is given by

$$\mathrm{proj}_{i,j} = \mathbf{n}_i \frac{(\mathbf{n}_i + \mathbf{n}_j^\star)\cdot(\mathbf{x}_i - \mathbf{x}_j)}{2\,\mathbf{n}_i\cdot(\mathbf{n}_i + \mathbf{n}_j^\star)}\,, \tag{10}$$

so this regularization step is computed as

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \sum_{j\in\mathcal{F}} W_i^{\lambda_c}(\mathbf{x}_j)\,\mathrm{proj}_{i,j}\,. \tag{11}$$

Regularization Along the Tangents: Similar to previous works, we insert repulsion forces [Turk 1991] to improve the distribution of the surface points by driving them along their tangent directions. At each surface point, we compute the weighted direction *away* from its neighbors, and displace it in this direction. This averaging is performed in a local neighborhood λ_f around the point. Explicitly, this regularization step is computed at each surface point as

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + 0.5\,\lambda_f \sum_{j\in\mathcal{F}} W_i^{\lambda_f}(\mathbf{x}_j)\,\mathrm{TN}_i(\mathbf{x}_j - \mathbf{x}_i)\,, \tag{12}$$

where TN_i projects onto the tangent plane of point i and then normalizes the result, and the 0.5 factor prevents two points from moving to the same location.

Insertion and Deletion: The last regularization step adds and deletes surface points according to changes in the underlying simulation. Points are deleted when the local point density is too high. We detect this by looking for pairs of points that are closer than ${}^3\!/_4\,\lambda_f$, in which case the most recently created point is deleted. Similarly, surface points are added when the local point density is

```
1 forall the surface points $i$ do
2     $\mathbf{n}_i \leftarrow \mathbf{g}_i$;
3     compute tangent $\mathbf{t}_i^1$ and bi-tangent $\mathbf{t}_i^2$, orthonormal to $\mathbf{n}_i$;
4     least square plane fitting in $\lambda_c$ to the frame $[\mathbf{t}_i^1, \mathbf{t}_i^2, \mathbf{n}_i]$;
5     $\mathbf{n}_i \leftarrow$ normal of plane;
6 forall the surface points $i$ do
7     $\mathbf{n}_i \leftarrow$ averaged normal in neighborhood $\lambda_c$;
8     $\mathbf{n}_i \leftarrow$ normalize $(\mathbf{n}_i)$;
```

Algorithm 1: Steps for computing the normals.

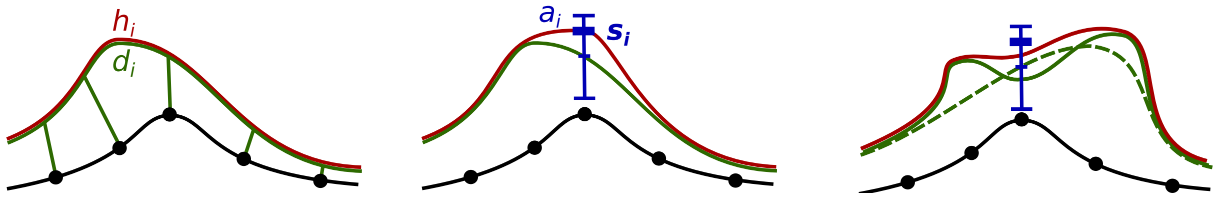

Figure 5: *Our wave seeding strategy. A wave moving from the left side of the simulation reaches the highlight region of the surface (left); seeding has yet to occur here, so the displayed (d_i, green) and internal (h_i, red) waves match. The underlying surface has high curvature at the center surface point (middle). We increase the oscillator amplitude (a_i) here from 0 to Δa, and compute a wave seeding value (s_i) from a cosine oscillator with amplitude a_i. We evolve the wave simulation and subtract the seeding values from the computed wave to obtain the new displaced wave value (right). The dashed green line shows what the displayed wave would have been if no wave seeding had occurred.*

too low, by Poisson disk sampling [Cook 1986]. For each point, we recompute the tangential direction as it was computed in the previous tangent regularization step. This generally points in the direction of lowest density, so we place a sphere of radius λ_f at a distance λ_f in this direction and search for neighboring points that fall inside the sphere. If none are found, the point density is too low and we create a point at the displaced sphere's center (Figure 4).

The three regularization steps are global operations that influence the *entire* surface. However, as they each consist of spatially local computations, we apply them several times to each animation frame until convergence. We used 30 iterations for the first frame and between 3 and 10 iterations per subsequent frame in all of our simulations. The number of iterations used for each scene is shown in Table 1. The accompanying video shows the uniform point density maintained by our surface operations on a deforming fluid surface.

3.5 Interactions with Obstacles

We have so far focused on the management of surface points in unobstructed flow, but special care must be taken when surface points approach obstacles. The regularization in Section 3.4 works best if the point distribution around each surface point is approximately uniform, which is not the case near obstacle boundaries.

Motivated by ghost and boundary [Schechter and Bridson 2012; Akinci et al. 2012] particles in SPH, we add ghost points for each surface point located close to an obstacle by reflecting the neighboring points of the current point w.r.t. the obstacle. We then simply apply the regularization steps to both "real" and ghost point sets.

4 Turbulence Creation and Evolution

Section 3 detailed the creation of high-density point sets that represent the underlying coarse fluid simulation surface. We can now add turbulent details atop this surface, simulating waves on its points.

4.1 Curvature Evaluation

Surface wave details should appear in areas of high activity, e.g., merging or separating regions. Mean curvature is a good indicator of these regions [Kim et al. 2013], as high curvature tends to denote under-resolved areas in the base simulation. Second-order neighborhood fitting is commonly used to compute curvature on point surfaces: this works well for smooth surfaces [Wang et al. 2013], but we deal with turbulent surfaces where quadratic fits can generate extreme and noisy curvatures, leading to instabilities over time.

We instead propose a new, robust alternative for evaluating curvature on point clouds. The curvature c_i at point i is defined as the signed distance of its neighbors to its tangent plane, easily obtained from the normal computed at each point (Section 3.4), which gives:

$$c_i = \sum_{j \in \mathcal{F}} W_i^{\lambda_c}(\mathbf{x}_j)\,(\mathbf{n}_i \cdot (\mathbf{x}_i - \mathbf{x}_j))\,. \tag{13}$$

While not identical to mean curvature, this measure is a very reliable criterion for wave generation. Moreover, thresholding any such criterion in a meaningful way across discretizations is very important. We derive practical thresholds $\{c_{\min}, c_{\max}\}$, evaluating Equation 13 at two extremal scenarios: single drops and thin sheets. For a surface of infinite point density at distance λ_c from the coarse particles, our measure evaluates to $c_{\max} = 0.15\,\lambda_c$ for a single drop and $c_{\min} \approx 0.077\,\lambda_c$ for a thin sheet. These thresholds (see Appendix A for derivations) are useful for parametrizing simulations.

4.2 Turbulence Creation

The simplest way to add turbulence to a surface is to add it directly to the wave heights. This can work well for grid-based methods [Kim et al. 2013] where surface curvature varies smoothly with respect to grid size, but particle-based methods often contain large, abrupt curvature variations. For instance, when a single particle falls onto a flat water surface, a discontinuous wave seeding causes an abrupt visual change in height. Curvature is always high on isolated droplets, causing uniform wave seeding over the droplet and unrealistic (non mass conserving) pulsing (see supplemental video).

A simple solution used in mesh-based simulations [Yu et al. 2012] is to turn off the wave seeding in regions of high curvature. This reduces artifacts from curvature discrepancies, but still produces abrupt changes in regions of near-maximum curvature. Moreover, it removes some of the waves caused by fine splashes characteristic of particle-based fluids, and limits the amount of new added details.

We propose a different approach, outlined in Algorithm 2 and illustrated in Figure 5. In lines 2 and 3, we throttle seeding in high

Data:

c_i	: surface curvature	c	: wave speed
$c_{\text{min,max}}$	: curvature thresholds	f_b	: base seeding frequency
a_i	: oscillator amplitude	f_o	: # of frequency octaves
Δa	: oscillator amplitude step size	W	: max. wave amplitude
A	: max. oscillator amplitude	F	: max. wave frequency
h_i	: internal wave height	t	: simulation time
v_i	: internal wave velocity	Δh_i	: laplacian
d_i	: displayed wave height	s_i	: seed value

```
1  forall the surface points i do
2      $a_{\text{tmp}} \leftarrow 2\,\text{smoothstep}(|c_i|, c_{\min}, c_{\max}) - 1$;
3      $a_i \leftarrow \text{clamp}(a_i + a_{\text{tmp}}\,\Delta a, 0, A)$;

4      $s_i \leftarrow 0$; $f \leftarrow f_b$; $a \leftarrow a_i$;
5      for $\lambda = 1 \,..\, f_o$ do
6          $s_i \leftarrow s_i + a_i\ \cos(t\,c\,f)$; $f \leftarrow 2\,f$; $a \leftarrow a/2$;
7      $h_i \leftarrow d_i + s_i$;
8  forall the surface points i do
9      $v_i \leftarrow v_i + c^2\,\Delta t\,\Delta h_i$; $h_i \leftarrow h_i + \Delta t\,v_i$; $d_i \leftarrow h_i - s_i$;
10     $d_i \leftarrow \text{clamp}(d_i, -W, W)$; $v_i \leftarrow \text{clamp}(v_i, -W\,F, W\,F)$;
```

Algorithm 2: Pseudocode for wave evolution and seeding.

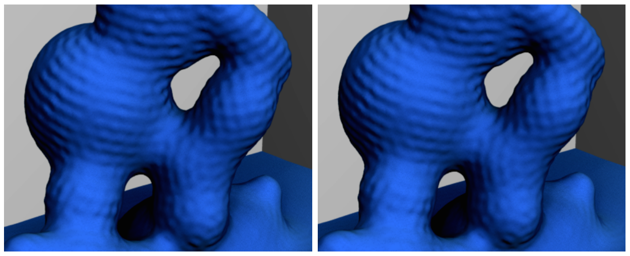

Figure 6: *Wave propagation with Laplace-Beltrami (left) and our flat Laplace operator (right). The approximation error is negligible: no observable differences even after* 1000 *simulation steps.*

curvature regions based on the curvature thresholds of Section 4.1. We seed with time-varying cosine oscillators (line 6), but these oscillations are never directly visualized. Instead, we apply the wave equation to them and only new waves that have *propagated out* from the cosine oscillations are visualized. This avoids double accumulation of wave values in regions of high curvature: once from their own oscillator and once from the waves generated by neighboring oscillators. This also causes waves to appear at the boundary of the seeding regions: for an isolated droplet, the seeding region has no boundary and, as no wave is seeded, the pulsing artifact is removed.

Our seeding is easy to implement: in addition to the wave value h_i used to solve the wave equation, we store a *displayed wave value* d_i. We store cosine oscillators separately in *seed values* s_i. At each step, we first add seed values to the displayed wave values to obtain the wave values. We perform wave simulation on h_i and subtract seed values from h_i to recover d_i. The d_i thus only contain new features propagated out from the seed values, in addition to features that have evolved from previous timesteps. We never visualize h_i.

Finally, inspired by smoke up-res methods [Kim et al. 2008], our approach seeds features across multiple frequency octaves. To enable further control, the base seeding frequency f_b of the cosine oscillators, and the number of additional octaves f_o, are exposed as user parameters. The accompanying video shows a comparison of our seeding method versus more direct approaches.

4.3 Turbulence Evolution

We evolve waves on the surface using the standard wave equation,

$$\partial_t^2 h = c\,\Delta h\,, \tag{14}$$

where c is the wave speed and Δ is the Laplace-Beltrami operator. We solve this equation explicitly with a symplectic Euler scheme that also requires the wave velocity v_i to be stored at each surface point. The linear wave equation, due to its dispersive nature, only approximates capillary surface wave behavior. The wave speed c must thus be selected empirically, for instance by estimating the desired traveled distance of waves between two given frames. Despite this, the linear wave model is widely used [Thuerey et al. 2010; Chentanez and Müller 2010] and yields convincing results. Similar to these works, we add a small amount of diffusion to approximate wave dissipation, but this is not required to maintain stability.

Some previous works solve differential equations involving the Laplace-Beltrami operator on point surfaces [Liang et al. 2012; Liang and Zhao 2013], requiring the evaluation of surface curvature as well as second derivatives of the embedded function. In our case, the curvature of the surface is small compared to the length λ_f at which the wave equation operates, so we instead approximate the Laplace-Beltrami operator in Equation 14 with a simpler, flat Laplace operator. We avoid the metric tensor computations of Laplace-Beltrami, which are both costly and difficult to robustly evaluate on point surfaces. Figure 6 compares our flat Laplace operator to Laplace-Beltrami, showing results that are indistinguishable even after 1000 simulation steps. Here, the approximation was 6 times faster to compute and gave a maximum height difference of only 1.4%. Appendix B further validates our approximation.

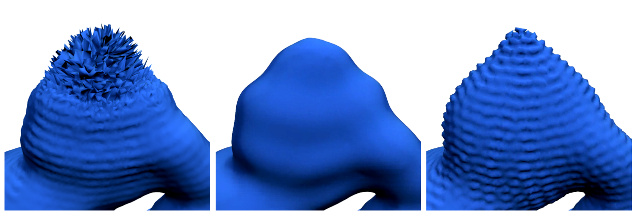

Figure 7: *Comparing the Laplacian computed with a least squares fit to our new discrete operator. When generating waves at a scale near the point density limit, the least squares fit fails to isolate the desired wavelength and becomes unstable (i.e., undesirable small scale waves, left). At the same wave frequency, our discrete Laplace operator correctly treats the wave (middle) and even remains stable when pushed to the Nyquist limit of the discretization (right).*

A common method [Liang and Zhao 2013] for computing the flat Laplace operator on a point surface is to use the derivatives of a local quadratic least squares approximation of the function. Although this works with densely sampled surfaces, we generate waves at scales that can be of the same order as the distance between points. Only a few points can then be used for the quadratic approximation, which is imprecise and leads to instabilities over time. Our supplemental video and Figure 7 illustrate such instabilities.

We instead compute the Laplace operator by using the tangent plane, which was previously obtained during surface normal computation (section 3.4). By projecting nearby points onto the tangent plane, the displacement defined by the wave values h_i becomes a function defined on the tangent plane. In this coordinate system, an affine approximation P of the function is first computed using standard least-square minimization. Subtracting the values of P on each neighboring point eliminates the zeroth- and first-order derivatives of the function, so the Laplacian can be evaluated directly as a weighted sum of discrete directional second-order derivatives:

$$\Delta h_i = \sum_{j\in\mathcal{F}} W_i^{2\lambda_f}(\mathbf{x}_j)\,\frac{4\left((h_j - P(\mathbf{x}_j)) - (h_i - P(\mathbf{x}_i))\right)}{||\mathbf{x}_i - \mathbf{x}_j||^2}\,. \tag{15}$$

We have found that a neighborhood radius size of $2\lambda_f$ works well. Appendix B shows how our operator approximates the Laplacian. To our knowledge, we are the first to introduce this discrete operator for computing the Laplacian on a meshless set of points. Our supplemental video and Figure 7 show that it matches computations using the usual quadratic least squares approximation. Furthermore, since it uses an affine least squares fit instead of a quadratic fit, our operator remains stable even for waves at the highest frequency representable by the surface points. As such, it is particularly well suited to our meshless point representations, and we observed it was 2 times faster to evaluate than the quadratic least squares Laplacian.

We inject ghost points when surface points approach obstacles (Section 3.5) and also use these ghost points during wave computations, where we simply copy the wave value on a ghost point from its original “real” surface point. This gives Neumann boundary conditions on the obstacles and yields the expected wave reflections.

5 Results and Discussion

Up-resed Simulations. We apply our method to coarse simulations generated using Houdini 13’s FLIP solver. We apply our method to a large 12.5 million particles input simulation (Figures 1 and 12). Even at this high resolution, we enhance the visual quality by generating waves on a 500K surface point representation.

Scene	# particles	# surface points	Total (sec/frame)	Advection	Regularization	Curvature	Laplacian	Wave	Disk I/O
Dam Break	12500k	500k	85.4	8.6%	53.1% (10)	5.4%	2.9%	0.3%	2.0%
River	400k	280k	42.2	14.7%	48.7% (3)	9.9%	10.8%	0.8%	15.1%
Double Drop	1400k	350k	25.9	8.1%	57.0% (5)	5.4%	7.9%	1.0%	4.9%
Stir	390k	17k	1.5	15.0%	49.9% (5)	6.7%	4.1%	1.1%	4.6%
	390k	66k	5.4	10.8%	58.1% (5)	7.5%	6.0%	0.8%	4.8%
	390k	145k	15.2	8.7%	61.0% (5)	7.0%	9.8%	0.6%	3.9%

Table 1: *Timings for the various steps of our algorithm. All performance statistics were computed on an Intel i7 quad core running at 3.4 GHz with 32GB of RAM. In the regularization column, the parenthesis indicate the number of regularization steps used per frame.*

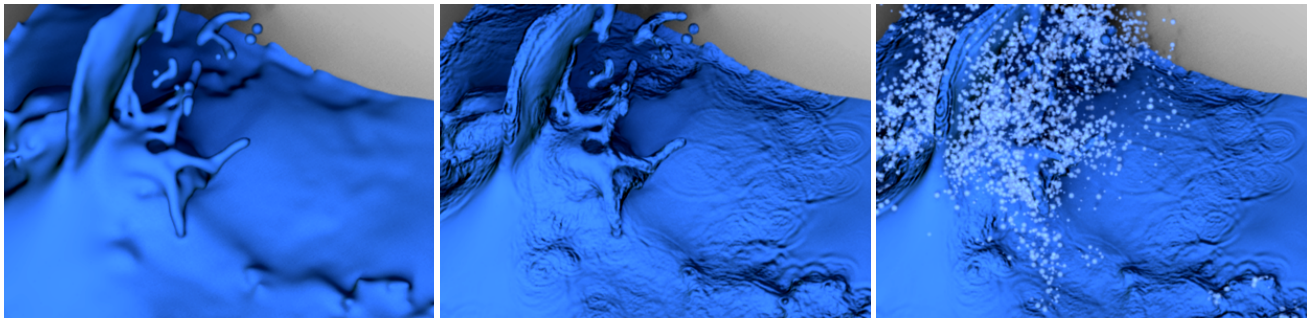

Figure 8: *We upres an input simulation (left, 1.4 million particles) with 400K surface points (middle), augmenting details over the bulk of the surface. We can also combine our results with other methods (i.e., [Ihmsen et al. 2012]) to further increase the visual fidelity (right).*

Our complete post-process requires 85.4s/frame, compared to the 241s/frame used for the input simulation. This illustrates our scalability beyond resolutions achievable with regular fluid solvers.

Figure 9 features a complex moving obstacle [Lait 2011], and demonstrates various levels of up-resing: from a 390K input coarse particle simulation, we generate 17K, 66K and 145K surface points. Each successive point set is able to resolve higher frequency details corresponding to successively higher visual fidelity. In contrast to previous work [Kim et al. 2013], our method does not require any additional information apart from the points used to represent the final surface to simulate a similar amount of wave detail.

Figure 11 shows a complex, turbulent riverbed. Even at 400K particles, the coarse simulation cannot capture the intense turbulence that characterizes a river's flow. Our up-resed output conveys this imagery, adding surface waves both where the water collides with rocks, as well as in stationary eddies behind these same obstacles.

Comparison with Full Resolution FLIP. Figure 10 compares the results of a high-resolution 4 million particles FLIP simulation with our method applied to a low resolution simulation comprising just 2500 particles. The 4 million particles are able to resolve certain fine structures, such as the splash, however, the final surface only contains rough, low frequency waves. In comparison, our method crisply resolves waves with many more frequencies. The 4 million particles scene requires 142s/frame while ours uses only 4.74s/frame, corresponding to a 30× speedup. Obtaining comparable waves with only a FLIP simulation would require even more particles, increasing our speedup for an equal-quality wave motion.

Implementation and Performance. Our method relies heavily on surface point lookups in small neighborhoods, so it is crucial to use an acceleration structure to store the surface point data. We used a hashing structure [Teschner et al. 2003] to improve the efficiency of these operations, yielding a speedup in the range of 20× for 27K surface points to 200× for 290K surface points compared to brute-force lookups. Most of the computations are performed on individual surface points, so trivial parallelizations using OpenMP further accelerated these operations by another 4 to 8×. Full computation breakdowns for our four scenes are provided in Table 1.

Limitations. Since we are designing an upres technique, we deliberately leave the coarse dynamics of the simulation untouched. Consequently, we do not reduce the size of the smallest underlying simulation structures. While this does not affect the majority of a simulation, fine isolated structures (i.e., droplets) are limited by the size of the input simulation. As mentioned in Section 4.2, seeding waves on isolated particles leads to visible mass loss and undesirable behavior, so we leave them untouched. However, our method remains compatible with techniques that directly address this problem, such as Ihmsen et al.'s approach [2012] (see Figure 8).

The timings in Table 1 show the majority of our computation is spent in surface regularization and neighborhood queries. This is as expected since the λ_c scale used for normal evaluation and regularization (Section 3.4), as well as for curvature computations (Section 4.1), does not decrease as the number of surface points increases. The number of neighbors thus grows quadratically with the total number of surface points. However, there are redundancies in these queries, since they all relate to the same underlying coarse simulation regardless of the final point count. A hierarchical method designed to instead select a constant-sized subset of representative neighbors for use in these queries would reduce the complexity of the regularization steps to that of all other steps. Designing such a selection method is a natural direction for future work.

6 Conclusion

We have presented a fully Lagrangian method for enhancing a particle-based liquid simulation using surface waves. This was

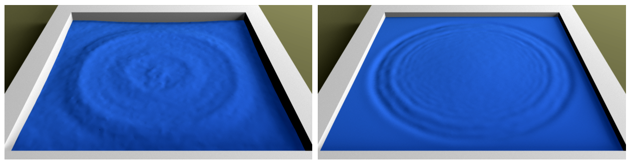

Figure 10: *Comparing a very high resolution (left; 4 million particles) and low resolution FLIP simulation up-resed with our method (right; 2500 coarse particles). The high resolution simulation only contains shallow, low frequency surface waves. Our result is up-resed to 15500 surface points, yielding much crisper surface waves.*

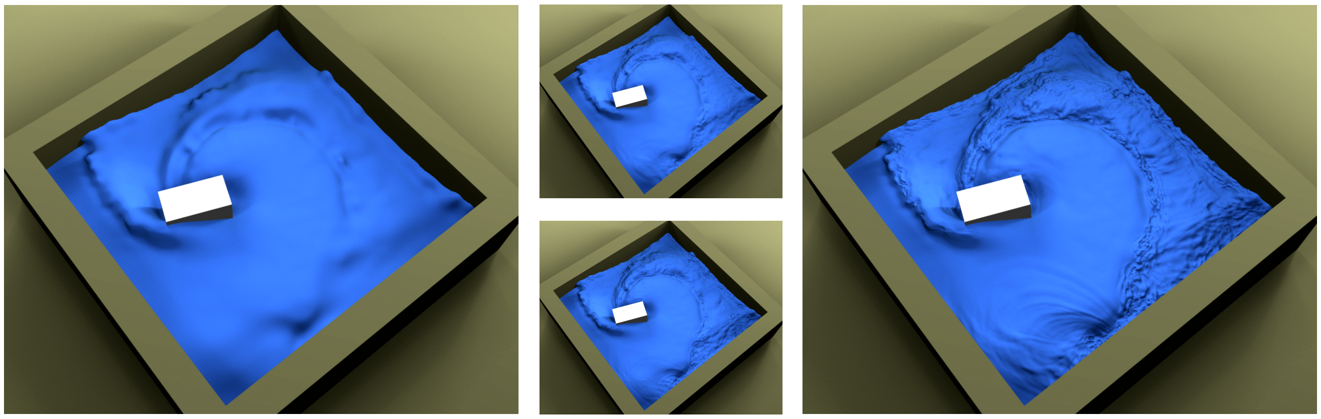
Figure 9: *We upres a 380K FLIP simulation (left) with 17K (middle, top), 66K (middle, bottom), and 145K (right) surface points.*

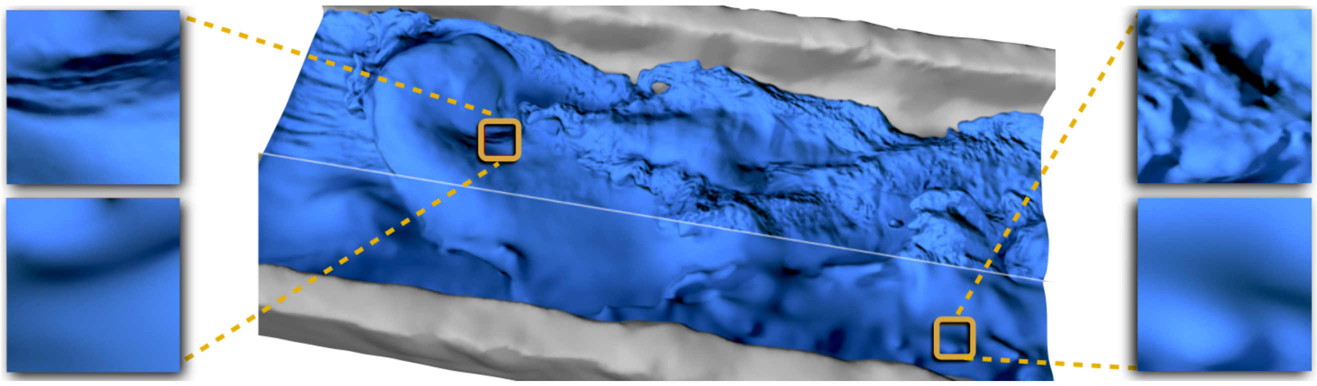
Figure 11: *We upres an input 400K particle FLIP simulation (middle bottom half; zoom-ins bottom) with 280K surface points (middle top half; zoom-ins top). Our surface waves interact realistically with the turbulent flow over the rocks and the resulting stationary eddies.*

made possible using a combination of several novel techniques, including a robust method for point surface creation and maintenance and a stable discrete Laplace operator, both of which can apply more broadly in settings involving surface operations on animated point sets. We also proposed a novel wave injection strategy based on bands of oscillators, and we have demonstrated that our method can efficiently process coarse input simulations (of both low- and high-resolutions) into highly detailed and turbulent liquid surfaces.

In the future, we plan to explore more complex and expressive waves models, art-directable editing controls for the fine-scale details, closer coupling to secondary particle systems for drops and foam effects, and the application of our method to other types of secondary surface simulations for Lagrangian data.

Appendix A: Threshold Computations

To evaluate our curvature measure on a surface of infinite point density, we consider the integral form of (13) as $|\mathcal{S}| \to \infty$. Here, surface points all have equal density ρ, so we can replace W with K. We omit the neighborhood size λ_c from the kernel notation for brevity. We work in a local frame with the surface point of interest at $(0, \lambda_c, 0)$ with normal $(0, 1, 0)$ and xz-tangent plane, yielding

$$c_i = \int_{\mathbf{x}\in S} K((0,\lambda_c,0) - \mathbf{x}\cdot\mathbf{e}_y)\,(\lambda_c - y)\,dS \Big/ \int_{\mathbf{x}\in S} K((0,\lambda_c,0) - \mathbf{x})\,dS\,.$$

For the case of a single drop, we consider the surface $\{x^2 + y^2 + z^2 = \lambda_c^2\}$. Solving the integral analytically in spherical coordinates $p(\theta,\phi) = (\lambda_c \sin\theta\cos\phi, \lambda_c\cos\theta, \lambda_c\sin\theta\sin\phi)$ yields

$$c_i = \frac{\int_{\phi=0}^{2\pi}\int_{\theta=0}^{\pi/3} \lambda_c\,(1-\cos(\theta))\,K((0,\lambda_c,0) - p(\theta,\phi))\,d\theta\,d\phi}{\int_{\phi=0}^{2\pi}\int_{\theta=0}^{\pi/3} K((0,\lambda_c,0) - p(\theta,\phi))\,d\theta\,d\phi} = \frac{3\lambda_c}{20}.$$

For the case of a thin sheet, we consider the surface

$$(\{y \geq 0\} \cap \{x^2 + y^2 = \lambda_c^2\}) \cup (\{y \leq 0\} \cap \{|z| = \lambda_c\}).$$

Since K is non-zero only in the cylindrical part of the thin sheet, we can use cylindrical coordinates $p(x,\theta) = (x, \lambda_c\cos\theta, \lambda_c\sin\theta)$:

$$c_i = \frac{\int_{x=-\lambda_c}^{\lambda_c}\int_{\theta=-\pi/3}^{\pi/3} \lambda_c\,(1-\cos(\theta))\,K((0,\lambda_c,0) - p(x,\theta))\,\lambda_c\,dx\,d\theta}{\int_{x=-\lambda_c}^{\lambda_c}\int_{\theta=-\pi/3}^{\pi/3} K((0,\lambda_c,0) - p(x,\theta))\,\lambda_c\,dx\,d\theta},$$

which evaluates numerically to $0.0771413\,\lambda_c$ (to double precision).

Appendix B: Laplace-Beltrami Approximation

We justify our Laplace approximation and detail computations used for Figures 6 and 7.

Flat Laplace Computation

First, we show that (15) indeed approximates the Laplace operator at surface point i. We orient the coordinate system to have origin at point i and tangent xy-plane. We express $W_i^{2\lambda_f}$ recentered at

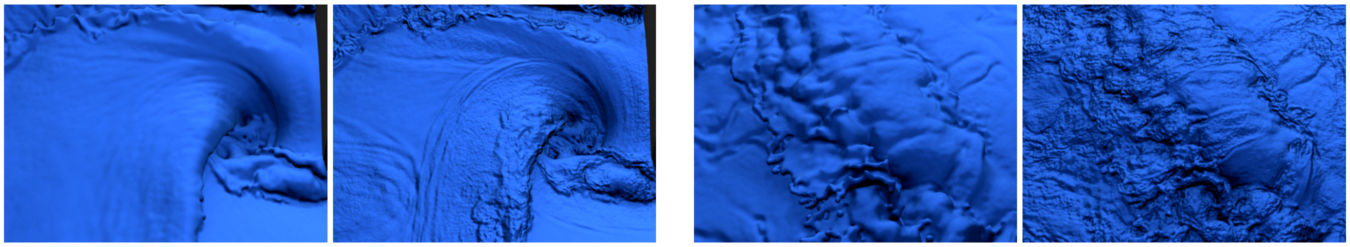
Figure 12: *Two close-ups of the Dam Break scene (Figure 1). In each pair, the left image shows the input simulation and the right image shows our upresed output surface. Even with 12M input particles, the input simulation fails to resolve finer surface details, so our method is still capable of significantly increasing the visual quality of the result even with high-resolution inputs.*

the origin as W, and override $\mathcal{F}$ as the neighboring points in those coordinates. We can rewrite the discrete operator as

$$\begin{aligned}
&\sum_{\mathbf{x}\in\mathcal{F}} 4\,W(\mathbf{x})((h(\mathbf{x}) - P(\mathbf{x})) - (h(\mathbf{0}) - P(\mathbf{0}))\Big/||\mathbf{x}||^2 \\
&= \sum_{\mathbf{x}\in\mathcal{F}} 4\,W(\mathbf{x})(h(\mathbf{x}) - P(\mathbf{0}) + \nabla P(\mathbf{0})\cdot\mathbf{x} - h(\mathbf{0}) + P(\mathbf{0}))\Big/||\mathbf{x}||^2 \\
&= \sum_{i\in\mathcal{F}} 4\,W(\mathbf{x})(h(\mathbf{x}) - \nabla h(\mathbf{0}) + O(\lambda_f^2)\cdot\mathbf{x} - h(\mathbf{0}))\Big/||\mathbf{x}||^2 \qquad (16) \\
&= \sum_{\mathbf{x}\in\mathcal{F}} \left(4\,W(\mathbf{x})(h(\mathbf{x}) - \nabla h(\mathbf{0})\cdot\mathbf{x} - h(\mathbf{0}))\Big/||\mathbf{x}||^2\right) + O(\lambda_f) \qquad (17)
\end{aligned}$$

where the error term in (16) results from the superconvergence demonstrated in Appendix A in Liang's work [Liang 2013], and is possible due to the approximate point distribution symmetry maintained by our method (Section 3.4). As the point density approaches infinity, the operator converges to

$$\iint_{\mathcal{D}} 4\,K(\mathbf{x})(h(\mathbf{x}) - \nabla h(\mathbf{0})\cdot\mathbf{x} - h(\mathbf{0}))\Big/||\mathbf{x}||^2\,d\mathbf{x} + O(\lambda_f) \qquad (18)$$

where $\mathcal{D}$ is the disk $x^2 + y^2 \leq (2\lambda_f)^2$ and K is the kernel $K_i^{2\lambda_f}$ recentered at the origin and normalized to 1. Again, note that (18) is only correct if the point density of the surface is uniform, which we maintain in our method. Using a Taylor expansion of h in the direction $\mathbf{x}$, we have $h(\mathbf{x}) = h(\mathbf{0}) + ||\mathbf{x}||h'_{\mathbf{x}}(\mathbf{0}) + \frac{1}{2}||\mathbf{x}||^2 h''_{\mathbf{x}}(\mathbf{0}) + O(||\mathbf{x}||^3)$, where $h'_{\mathbf{x}}$ and $h''_{\mathbf{x}}$ are the first and second directional derivatives of h w.r.t. $\mathbf{x}$. Substituting the Taylor expansion into (18), we arrive at

$$\begin{aligned}
&= \iint_{\mathcal{D}} 4\,K(\mathbf{x})\left(\frac{1}{2}||\mathbf{x}||^2 h''_{\mathbf{x}}(\mathbf{0}) + O(||\mathbf{x}||^3)\right)\Big/||\mathbf{x}||^2\,d\mathbf{x} + O(\lambda_f) \\
&= \iint_{\mathcal{D}} 2\,K(\mathbf{x})h''_{\mathbf{x}}(\mathbf{0})\,d\mathbf{x} + O(\lambda_f). \qquad (19)
\end{aligned}$$

The error term disappears since λ_f approaches zero as the point density increases. We split (19), effect a clockwise rotation of $90^{\deg}$, and use the symmetry of K to obtain

$$\begin{aligned}
&= \iint_{\mathcal{D}} K(\mathbf{x})h''_{\mathbf{x}}(\mathbf{0})\,d\mathbf{x} + \iint_{\mathcal{D}} K(\mathbf{x})h''_{\mathbf{x}}(\mathbf{0})\,d\mathbf{x} \qquad (20) \\
&\approx \iint_{\mathcal{D}} K(x,y)h''_{(x,y)}(\mathbf{0})\,d\mathbf{x} + \iint_{\mathcal{D}} \underbrace{K(-y,x)}_{=K(x,y)} h''_{(-y,x)}(\mathbf{0})\,d\mathbf{x} \qquad (21) \\
&= \iint_{\mathcal{D}} K(x,y)\left(h''_{(x,y)}(\mathbf{0}) + h''_{(-y,x)}(\mathbf{0})\right)d\mathbf{x} \qquad (22) \\
&= \iint_{\mathcal{D}} K(x,y)\Delta h(\mathbf{0})\,d\mathbf{x} = \Delta h(\mathbf{0}) \iint_{\mathcal{D}} K(x,y)\,d\mathbf{x} = \Delta h(\mathbf{0}) \qquad (23)
\end{aligned}$$

where the equality between (22) and the left most equation in (23) leverages the invariance of the Laplacian under rigid deformation.

Flat Surface Approximation

We can now justify our claim in Section 4.3 that locally approximating the curved surface with a flat surface yields negligible errors in the evaluation of differential quantities. We parameterize a quadratic surface q about a given surface point as

$$q(x,y) = q_{00} + q_{10}x + q_{01}y + q_{20}x^2 + q_{11}xy + q_{02}y^2. \qquad (24)$$

Following [Liang and Zhao 2013], we compute a least square quadratic approximation centered at the surface point for both the surface (f) and the wave function (h) to approximate the Laplace-Beltrami operator at this point. The full expression for the operator can be derived as

$$\frac{\left(\begin{array}{l}
2\left(1 + f_{01}^4 + 2f_{01}^2 + f_{10}^2 + f_{10}^2 f_{01}^2\right)h_{02} \\
+2\left(1 + f_{10}^4 + 2f_{10}^2 + f_{01}^2 + f_{10}^2 f_{01}^2\right)h_{20} \\
+\left(\begin{array}{c} f_{20}f_{01}^3 + 3f_{01}^3 f_{02} + 3f_{01}f_{02} \\ +2f_{10}^2 f_{01}f_{02} + f_{01}f_{20} + 2f_{10}^2 f_{01}f_{20} \\ +2f_{10}f_{11} + f_{10}^3 f_{11} + 3f_{10}f_{01}^2 f_{11} \end{array}\right)h_{01} \\
+\left(\begin{array}{c} f_{02}f_{10}^3 + 3f_{10}^3 f_{20} + 3f_{10}f_{20} \\ +2f_{10}f_{01}^2 f_{20} + f_{10}f_{02} + 2f_{10}f_{01}^2 f_{02} \\ +2f_{01}f_{11} + f_{01}^3 f_{11} + 3f_{10}^2 f_{01}f_{11} \end{array}\right)h_{10} \\
+\left(2f_{10}f_{01} + 2f_{10}^3 f_{01} + 2f_{10}f_{01}^3\right)h_{11}
\end{array}\right)}{1 + f_{01}^2 + f_{10}^2} \qquad (25)$$

Notice that if we align our local coordinate system with the surface, i.e., $f_{10} = f_{01} = 0$, (25) simplifies to $2h_{20} + 2h_{02}$, which is the flat Laplace operator. While our normal computation in Algorithm 1 attempts to get as close as possible to this perfect alignment, numerical errors will be present. Still, Figure 6 shows that f_{10} and f_{01} are small enough to permit our approximation with a flat Laplacian. Moreover, by only keeping the first-order terms of (25), i.e., ignoring terms that depend at least quadratically on f_{10} and f_{01}, we arrive at the first-order approximation

$$\begin{aligned}
2h_{02} + 2h_{20} &+ \left(3f_{01}f_{02} + f_{01}f_{20} + 2f_{10}f_{11}\right)h_{01} \\
&+ \left(3f_{10}f_{20} + f_{10}f_{02} + 2f_{01}f_{11}\right)h_{10}.
\end{aligned} \qquad (26)$$

We apply this expression in Figure 6 when computing the Laplace-Beltrami operator, where $(2h_{02} + 2h_{20})$ is evaluated using (15) and the wave function derivatives h_{10} and h_{01} are evaluated using the affine approximation P already computed in (15).

Acknowledgments

We acknowledge the following support: NSERC CGS-D (Mercier), NSERC CGS-M (Beauchemin), NSF CAREER Award IIS-1253948 (Kim) and NSERC Discovery Grant 418519-2012 (Nowrouzezahrai).

References

AKINCI, N., IHMSEN, M., AKINCI, G., SOLENTHALER, B., AND TESCHNER, M. 2012. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph. 31*, 4 (July), 62:1–62:8.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph. 9*, 1, 3–15.

AUTODESK, 2014. Bifröst for Autodesk Maya. `http://www.autodesk.com/products/maya/overview`.

BHATACHARYA, H., GAO, Y., AND BARGTEIL, A. 2011. A level-set method for skinning animated particle data. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 17–24.

BLINN, J. F. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (July), 235–256.

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

BUDSBERG, J., LOSURE, M., MUSETH, K., AND BAER, M. 2013. Liquids in "The Croods". In *ACM SIGGRAPH Digital Production Symposium (DigiPro)*.

CHENTANEZ, N., AND MÜLLER, M. 2010. Real-time simulation of large bodies of water with small scale details. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

COOK, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1 (Jan.), 51–72.

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

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

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

GUENNEBAUD, G., GERMANN, M., AND GROSS, M. H. 2008. Dynamic sampling and rendering of algebraic point set surfaces. *Comput. Graph. Forum 27*, 2, 653–662.

HUANG, R., MELEK, Z., AND KEYSER, J. 2011. Preview-based sampling for controlling gaseous simulations. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 177–186.

IHMSEN, M., AKINCI, N., AKINCI, G., AND TESCHNER, M. 2012. Unified spray, foam and air bubbles for particle-based fluids. *The Visual Computer 28*, 6-8, 669–677.

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

JEONG, S., AND KIM, C. 2013. Combustion waves on the point set surface. *Comput. Graph. Forum 32*, 7, 225–234.

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

KIM, T., THUEREY, N., JAMES, D., AND GROSS, M. 2008. Wavelet turbulence for fluid simulation. In *ACM Trans. Graph.*

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

LAIT, J. 2011. Correcting low frequency impulses in distributed simulations. In *ACM SIGGRAPH Talks*, 53:1–53:2.

LIANG, J., AND ZHAO, H. 2013. Solving partial differential equations on point clouds. *SIAM J. Sci. Comput. 35*, 3.

LIANG, J., LAI, R., WONG, T. W., AND ZHAO, H. 2012. Geometric understanding of point clouds using laplace-beltrami operator. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

MACDONALD, C. B., MERRIMAN, B., AND RUUTH, S. J. 2013. Simple computation of reaction–diffusion processes on point clouds. *P. Natl. Acad. Sci. 110*, 23, 9209–9214.

MACKLIN, M., MÜLLER, M., CHENTANEZ, N., AND KIM, T.-Y. 2014. Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July), 153:1–153:12.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *ACM SIGGRAPH/Eurographics Symp. on Computer animation*, 154–159.

NARAIN, R., SEWALL, J., CARLSON, M., AND LIN, M. C. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph. 27* (December), 166:1–166:8.

NEXT LIMIT TECHNOLOGIES, 2014. RealFlow. `http://www.realflow.com/`.

NIELSEN, M. B., AND BRIDSON, R. 2011. Guide shapes for high resolution naturalistic liquid simulation. *ACM Trans. Graph.*.

NIELSEN, M. B., CHRISTENSEN, B. B., ZAFAR, N. B., ROBLE, D., AND MUSETH, K. 2009. Guiding of smoke animations through variational coupling of simulations at different resolutions. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*

OSHER, S., AND FEDKIW, R. 2003. *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag, New York.

PAN, Z., HUANG, J., TONG, Y., ZHENG, C., AND BAO, H. 2013. Interactive localized liquid motion editing. *ACM Trans. Graph.*.

SCHECHTER, H., AND BRIDSON, R. 2008. Evolving sub-grid turbulence for smoke animation. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 1–7.

SCHECHTER, H., AND BRIDSON, R. 2012. Ghost sph for animating water. *ACM Trans. Graph. 31*, 4 (July), 61:1–61:8.

SHAO, X., ZHOU, Z., ZHANG, J., AND WU, W. 2014. Realistic and stable simulation of turbulent details behind objects in smoothed-particle hydrodynamics fluids. *Computer Animation and Virtual Worlds*.

SHI, L., AND YU, Y. 2005. Taming liquids for rapidly changing targets. *ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*.

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

TESCHNER, M., HEIDELBERGER, B., MUELLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of Vision, Modeling, Visualization*, 47–54.

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

THUEREY, N., WOJTAN, C., GROSS, M., AND TURK, G. 2010. A Multiscale Approach to Mesh-based Surface Tension Flows. *ACM Transactions on Graphics (SIGGRAPH) 29 (4)* (July), 10.

THUEREY, N., KIM, T., AND PFAFF, T. 2013. Turbulent fluids. In *ACM SIGGRAPH 2013 Courses*, 6:1–6:1.

TURK, G. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. In *Proceedings of SIGGRAPH*, 289–298.

WANG, H., MILLER, G., AND TURK, G. 2007. Solving general shallow wave equations on surfaces. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*

WANG, R., YANG, Z., LIU, L., AND CHEN, Q. 2013. Discretizing laplace–beltrami operator from differential quantities. *Communications in Mathematics and Statistics 1*, 3, 331–350.

WILLIAMS, B. W. 2008. Fluid surface reconstruction from particles. *M.S. Thesis, The University of British Columbia, Canada*.

WOJTAN, C., MÜLLER-FISCHER, M., AND BROCHU, T. 2011. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses*, 8:1–8:84.

YU, Q., NEYRET, F., BRUNETON, E., AND HOLZSCHUCH, N. 2009. Scalable real-time animation of rivers. *Comput. Graph. Forum 28*, 2, 239–248.

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

YUAN, Z., ZHAO, Y., AND CHEN, F. 2012. Incorporating stochastic turbulence in particle-based fluid simulation. *The Visual Computer 28*, 5, 435–444.

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

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *ACM Trans. Graph.*, vol. 24(3), ACM, 965–972.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop 3d: an interactive system for point-based surface editing. In *ACM Trans. Graph.*, vol. 21(3), ACM, 322–329.