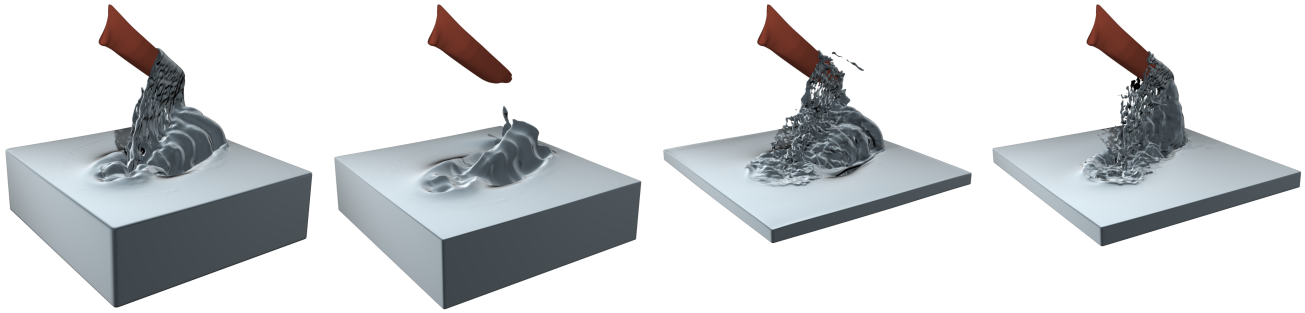
Guide Shapes for High Resolution Naturalistic Liquid Simulation

Michael B. Nielsen*
Weta Digital

Robert Bridson†
University of British Columbia
Weta Digital



(a) (b) (c) (d)

Figure 1: *A boat emerges from water.* ***(a)*** *Adequate depth is needed for the desired large-scale disturbances.* ***(b)*** *We compute a* guide shape *from the finalized coarse solve to capture the deep motion.* ***(c)*** *The guide shape constrains a high resolution simulation of a thin outer shell of liquid to keep the same look.* ***(d)*** *A high resolution simulation in shallow water fails to capture the large-scale motion.*

Abstract

Art direction of high resolution naturalistic liquid simulations is notoriously hard, due to both the chaotic nature of the physics and the computational resources required. Resimulating a scene at higher resolution often produces very different results, and is too expensive to allow many design cycles. We present a method of constraining or *guiding* a high resolution liquid simulation to stay close to a finalized low resolution version (either simulated or directly animated), restricting the solve to a thin outer shell of liquid around a *guide shape*. Our method is generally faster than an unconstrained simulation and can be integrated with a standard fluid simulator. We demonstrate several applications, with both simulated and hand-animated inputs.

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

Keywords: animation, fluid modeling, fluid simulation, physically based animation, constructive solid geometry

Links: DL PDF

1 Introduction

A common problem for liquid simulation in film is the high computational cost, both in time and memory, of high resolution simulation. Even if it is possible to simulate at the required resolution, the many iterations needed to achieve the desired artistic result (varying initial and boundary conditions, parameters, etc.) may still be infeasible. While most of the design work can ideally be done at low resolution, liquid dynamics are chaotic enough that later increasing the resolution often significantly changes the overall look and timing. This is due to numerous factors such as numerical viscosity, fidelity of solid geometry on the grid, when topological changes occur etc. To reduce the number of costly iterations at high resolution, we desire a way to guide the high resolution simulation to more closely follow the finalized low resolution version, while adding natural-looking extra detail. Note that our focus is entirely on naturalistic scenarios, not supernatural effects; art direction may nevertheless demand subtly nonphysical behavior, *e.g.* timing a splash to music, which further complicates pure simulation.

We introduce **guide shapes** in response. High resolution is often only necessary for small details at the surface and for splashes, while low resolution suffices for the deeper flow—*e.g.* ocean wave disturbances decay exponentially with depth and wave number [Bridson 2008]. Therefore we take the deeper flow from a finalized low resolution simulation or even a hand-crafted pre-visualization animation. Our method extracts a guide shape offset below the surface of the input, creates a matching velocity field throughout the volume if one is not given, determines an appropriate volume for seeding liquid in just a surface layer for the high resolution guided simulation, and imposes the guide shape as a boundary constraint on that layer (Figure 1). The high resolution version is then faster and stays closer to the desired result. Though some experimentation is still necessary to obtain the desired extra detail, our approach significantly reduces the number and expense of design iterations required at high resolution.

We have implemented our method as a plug-in to a commercially available fluid solver, Naiad, and have successfully tested the workflow with artists in feature film production. We include here several examples illustrating improved correspondence between low and high resolution, artistic control, and faster final simulations.

2 Related Work

Fluid control was introduced to graphics by Foster and Metaxas [1997]. Several authors have since addressed the problem of match-

*e-mail: mnielsen@wetafx.co.nz
†e-mail: rbridson@cs.ubc.ca

ing predefined key frames [Treuille et al. 2003; McNamara et al. 2004; Hong and Kim 2004] or moving target shapes [Fattal and Lischinski 2004; Shi and Yu 2005a; Shi and Yu 2005b]. Whereas these methods closely match the liquid surface to an animated surface or individual key frames with supernatural effects in mind, we impose control at an interface embedded inside the liquid for a more natural-looking surface flow. Furthermore we simulate only in a relatively thin outer shell.

Rasmussen *et al.* directed liquid simulations [2004] with particles that define soft or hard control on the flow, via blending or boundary conditions. Sachs *et al.* [2010] combined Rasmussen *et al.*'s methods with those of Shi *et al.* [2005a] to better control liquids. These authors all used a Neumann pressure boundary condition as we do to enforce hard velocity constraints in certain regions of the fluid. Shi *et al.* also used a divergence-free force field based on potential flow; we similarly investigate potential flow for defining velocities on the guide shape and in liquid emission, and compare this to simpler velocity extrapolation.

Thürey *et al.* [2006] proposed a particle-based framework for controlling low frequency flow components but still simulated the entire liquid as opposed to a thin shell. Horvath *et al.* [2009] later used particles to control fire. Several examples of liquid characters have also been documented [Wiebe and Houston 2004; Trojansky 2008]. In the context of naturalistic controlled flow, Mihalef *et al.* [2004] animated and controlled breaking waves by combining art direction in 2D with full simulation in 3D.

We also tackle the efficiency of high resolution simulation. Adaptive grids are the natural choice if efficiency is the sole issue, using coarse cells deeper in the flow: Losasso *et al.* [2004] used an octree while Irving *et al.* [2006] employed tall cells and an RLE data structure. These methods captured two-way interaction between the coarse and fine parts of the liquid, which our method does not, but the time stepping of the coarse cells is limited by the CFL condition in the fine cells. Additionally, our method improves the correspondence between low and high resolution which is neither the focus nor a property of the adaptive methods.

Our approach is also related to methods which add extra detail to bulk fluid flow. Vortex particle and turbulence synthesis methods [Selle et al. 2005; Kim et al. 2008] add detail by increasing vorticity while simulating or as a post process, to some extent also ensuring a correspondence between the bulk flow and the enhanced flow. However, these methods do not capture the full 3D physics of a liquid surface at higher resolution. For smoke, Nielsen *et al.* [2009] improved the correspondence between low and high resolution simulations, but their method is not directly applicable to liquids, nor does it help with efficiency. Patel *et al.* [2009] coupled a high resolution 2D simulation to an existing low resolution 3D simulation, using the 2D simulation to displace and add more detail. Similarly Angst *et al.* [2008] used the shallow water equations to enhance a liquid surface mesh. Neither of these methods can add fully 3D higher resolution surface effects such as small waves breaking.

3 The Guide Shape Method

The input to our method is a shape in level set representation with a velocity field defined either at the surface or everywhere inside the volume. For input from a low resolution simulation a volumetric velocity field will usually be available, and for artist-animated input the surface velocity field can be computed by finite differences on point correspondences. The core of our guiding method requires a volumetrically defined velocity field, thus in the latter case we interpolate the surface velocities throughout the volume. The final output is a guided liquid, which generally matches the input shape to some extent but adds additional detail and motion.

Symbol	Volume
Ω_I	Input Volume
Ω_L	Guided Liquid Volume
Ω_P	Preliminary Guiding Volume
Ω_G	Final Guiding Volume
Ω_R	Reseeding Volume
Ω_B	Union of Guided Liquid and Guiding Volume
Ω_S	Volume of Kinematic Solids
Ω_M	Mask Volume

Table 1: *Definition of volumes used.*

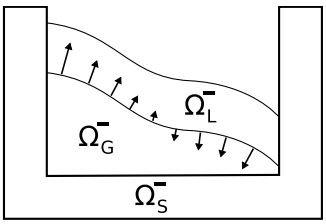


Figure 2: *Guided liquid volume Ω_L^-, guiding volume Ω_G^- and kinematic solids volume Ω_S^-.*

To clarify algorithmic details, we denote a volume by Ω and the matching level set function at time step m by ϕ^m, but omit the time step whenever it is clear from context. The surface of the volume is $\partial\Omega$, the interior is $\Omega^- = \{\mathbf{p} | \, \phi(\mathbf{p}) < 0\}$ and the exterior is $\Omega^+ = \{\mathbf{p} | \, \phi(\mathbf{p}) > 0\}$. Table 1 lists the specific volumes used throughout the text.

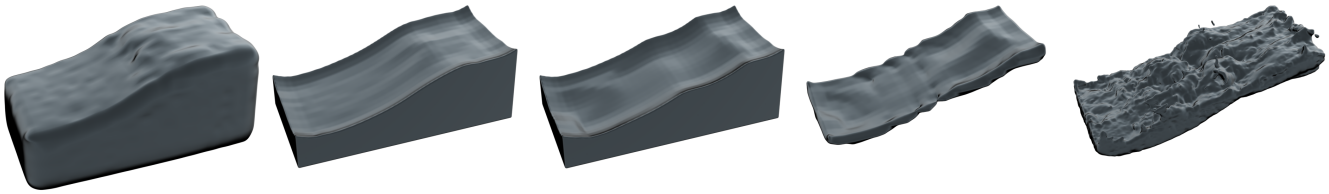
Referring to Figure 2, we enforce a velocity boundary condition on the guided liquid volume Ω_L^-, applied at the contact interface between the guided liquid and a guide volume Ω_G^- computed from both the input geometry and the evolving guided liquid. The equations of motion are solved only in the guided liquid volume, seeded inside a band of user-specified width close to the guide shape.

3.1 Velocity Boundary Condition

The rationale for the guide shape method is that in the true solution of the Navier-Stokes equations, one can replace an arbitrary moving volume of fluid with a velocity boundary condition at its surface (taking the surface velocity from the original flow) without changing the rest of the fluid.[1] Indeed, this is true for almost any continuum, under the usual Cauchy Postulate that inter-continuum forces are local contact tractions. Therefore if the low resolution input stays close to the correct physics at the guide depth, imposing the guide shape as a velocity boundary condition on the guided simulation will lead to the correct physical solution.

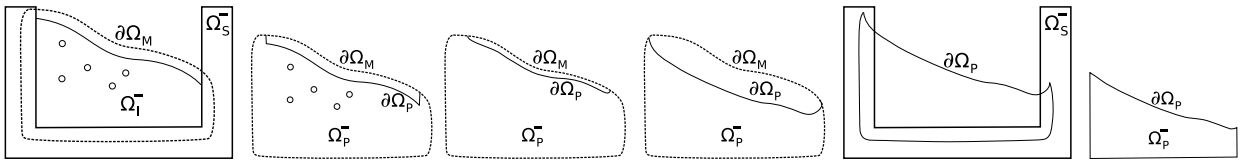
We guide the high resolution liquid volume Ω_L by imposing a velocity boundary condition at the interface with the guide shape, $\partial\Omega_G \cap \partial\Omega_L$, using the guide velocity defined there. We implement this by specifying the guide shape as a kinematic solid with inflow/outflow velocities [Bridson 2008]: in particular, note that the guide velocity is derived from the input velocity field, and may not match the motion or evolution of the guide shape geometry itself. The guided liquid may flow in or out of the guide shape, with the solver deleting particles in the former case, and our reseeding algorithm creating new liquid in the latter.

[1] One can equivalently impose a traction boundary condition, specifying pressure for an inviscid fluid, but a velocity boundary condition allows us to conveniently use existing functionality for kinematic solids.



(a) (b) (c) (d) (e)

Figure 3: ***(a)*** *Input shape $\partial\Omega_I$.* ***(b)*** *Preliminary guide $\partial\Omega_P$.* ***(c)*** *Final guide $\partial\Omega_G$.* ***(d)*** *Reseeding surface $\partial\Omega_R$.* ***(e)*** *Guided liquid $\partial\Omega_L$.*



(a) line 1 (b) line 2 (c) line 5 (d) line 7 (e) line 8 (f) line 9

Figure 4: *Illustrates the individual steps of Algorithm 1. References to Algorithm 1 are given by line numbers.* ***(a)*** *Construct mask for localizing computations.* ***(b)*** *Union solids and input shape (inside the mask region only).* ***(c)*** *Dilate and close air pockets.* ***(d)*** *Erode to ensure the desired simulation depth of guided liquid.* ***(e)*** *Intersect guiding shape with mask.* ***(f)*** *Subtract solids from guiding shape.*

Inside the solver, for inviscid fluids, the kinematic velocities are used in a Neumann boundary condition on pressure when setting up the Poisson linear system for pressure projection: $\frac{\partial p}{\partial \hat{\mathbf{n}}} = \frac{\rho}{\Delta t}(\tilde{\mathbf{u}} - \mathbf{u}_I) \cdot \hat{\mathbf{n}}$ where ρ is the density of the liquid, p is the pressure, Δt is the time step, $\mathbf{u}_I$ is the interpolated guiding velocity field and $\tilde{\mathbf{u}}$ is the velocity field after advection and adding forces using an operator splitting approach. Hence a constraint is enforced on the component of velocity normal to the kinematic solid. While our examples do not use viscous fluids, in the viscous case the kinematic velocities are similarly used as general velocity boundary conditions in the unsteady Stokes linear system.

3.2 Guide Shape Computation

The guide shape is built in two stages. First we compute a preliminary guide shape (Figure 3b) from the input geometry only (Figure 3a). This step is independent of the guided simulation and needs only to be performed once as a precomputation step for each input shape. In fact, the computation for each frame is independent and can thus be done in parallel, provided that input shapes for all frames are available. The second stage is performed *online*, *i.e.* interleaved with the guided liquid simulation (Figure 3e). In the second stage we compute the final guide shape (Figure 3c) from the guided liquid simulation and the precomputed preliminary guide shape. Following the online computation of the guide shape, we compute the reseeding volume (Figure 3d). Liquid will be reseeded inside the reseeding volume at each time step to prevent air pockets from appearing between the guide shape and the guided liquid.

It is essential to compute the final guide shape from both the preliminary guiding shape (which depends on the input geometry) and from the guided liquid simulation. Using the input geometry for the precomputed guide ensures that the final guide shape is fully enclosed in the volume of the input geometry and hence that the velocity field at the guide surface is well defined. On the other hand it is important to take the surface of the guided liquid into account as otherwise a minimum simulated depth would not be ensured, for example in the case of a splash only resolved at high resolution. In particular if this stage is omitted the guide shape would be able to protrude outside the guided liquid, leading to artifacts.

In the following descriptions of the algorithms we will be using $\Omega^{[a;b]} = \{\mathbf{p} | \, d(\mathbf{p}) \geq a \wedge d(\mathbf{p}) \leq b\}$ to denote a band portion of the domain where d is the signed distance function induced by ϕ.

3.2.1 Precomputing the Preliminary Guide

The preliminary guide shape is built with a sequence of level set operations (Figure 4). We first dilate the input shape by a small amount to form a mask for localizing all subsequent computations. We union the input geometry with selected kinematic solids—solids where it's not important to have high resolution interaction below the liquid-air surface (*e.g.* in the boat wake example, Figures 5 and 7, the container is included but not the boat itself since we want maximum detail around the boat). We then close any small gaps or air pockets through a combination of dilation and overwriting of small holes, redistance, and erode to get the desired simulation depth from the input shape. The precomputation stage is more formally given by Algorithm 1, and the individual steps of the algorithm apart from redistancing are depicted in Figure 4.

Algorithm 1 ϕ_P = ComputeGuide

Input: ϕ_I {input shape}
Input: ϕ_S {kinematic solids}
Input: α_M {mask Ω_M^- equals input geometry Ω_I^- dilated by α_M}
Input: α_b {final narrow band radius of ϕ_P}
Input: α_a {max radius of air pockets to be closed}
Input: α_d {depth of gaps to to be filled}
Input: α_e {minimum simulation depth of Ω_L}
Require: $(\alpha_d + \alpha_a) < \alpha_M$
Require: $\alpha_M < \alpha_b$
1: **dilate input to obtain mask, fig 4a** { $\phi_M = \phi_I - \alpha_M$ }
2: **union solids and input, fig 4b** $\{\phi_P = \min(\phi_S, \phi_I)$ on Ω_M^- }
3: **redistance** {Solve $|\nabla\phi_P| = 1$ on $\Omega_P^{[0;\alpha_d+\alpha_a]} \cap \Omega_M^-$}
4: **dilate to fill gaps** $\{\phi_P = \phi_P - \alpha_d$ on Ω_M^- }
5: **close air pockets, fig 4c** $\{\phi_P = -\alpha_a$ in holes less than $\alpha_a\}$
6: **redistance** {Solve $|\nabla\phi_P| = 1$ on $\Omega_P^{[-(\alpha_d+\alpha_e);0]} \cap \Omega_M^-$}
7: **erode, fig 4d** $\{\phi_P = \phi_P + (\alpha_d + \alpha_e)$ on $\Omega_M^-\}$
8: **intersect with mask, fig 4e** $\{\phi_P = \max(\phi_P, \phi_M)$ }
9: **subtract solids, fig 4f** $\{\phi_P = \max(\phi_P, -\phi_S)$ }
10: **redistance** {Solve $|\nabla\phi_P| = 1$ on $\Omega_P^{[-\alpha_b;\alpha_b]}$}

Note that redistancing between dilation and erosion is necessary to ensure gaps closed by the dilation are not subsequently reopened. To close air pockets specifically, our implementation uses a union-find algorithm to detect small holes (radius below α_a) surrounded by liquid and overwrites them with a negative value, prior to redistancing. Our implementation uses narrow band distance volumes, and solves the Eikonal equation in redistancing with a parallel fast marching implementation [Sethian 1996] localized inside Ω_M. At $\partial\Omega_M$ we impose a Neumann boundary condition $\partial\phi/\partial\hat{\mathbf{n}} = 0$ on the distance function to avoid an artificial zero-crossing there. Our typical values for the parameters are $\alpha_b = 2\Delta x$, $\alpha_a = 5\Delta x$, $\alpha_d = 2.5\Delta x$, $\alpha_M = \alpha_d + \alpha_a$ where Δx is the width of a grid cell. The value of α_e depends on the example at hand. When increasing the resolution, the parameter values are scaled correspondingly.

3.2.2 Computing the Final Guide

We run Algorithm 2 at the start of every time step of the simulation to compute the final guide shape. First we construct a bulk volume (lines 1–3 of Algorithm 2) as the union of the guided liquid at the current time step ϕ_L^n (Figure 3e) and the final guide shape from the previous time step advected forward to the current time step $\tilde{\phi}_G^n$ (Figure 3c from the previous time step advected forward). This volume corresponds to the bulk liquid volume as if we had simulated everywhere, without a guide shape. We next perform the same sequence of operations on this bulk volume as done by Algorithm 1 and illustrated by Figure 4 (line 4 of Algorithm 2). Finally we intersect this with the precomputed preliminary guide ϕ_P (line 5 of Algorithm 2). Algorithm 2 is bootstrapped in the first time step by setting $\phi_G^0 = \phi_P^0$.

Algorithm 2 ϕ_G = ComputeFinalGuide

Input: ϕ_P {preliminary guide}
Input: ϕ_G^{n-1} {guide at previous time step}
Input: ϕ_L {guided liquid}
Input: ϕ_S {kinematic solids}
Input: α_M {mask Ω_M^- equals input geometry Ω_I^- dilated by α_M}
Input: α_b {final narrow band radius of ϕ_P}
Input: α_a {max radius of air pockets to be closed}
Input: α_d {depth of gaps to to be filled}
Input: α_e {minimum simulation depth of guided liquid}
Input: Δt {time between time steps $n-1$ and n}
Require: $(\alpha_d + \alpha_a) < \alpha_M$
Require: $\alpha_M < \alpha_b$
1: **advect** ϕ_G^{n-1} **one time step forward in input velocity** $\mathbf{u}_I$ {Solve $\frac{\partial\phi_G}{\partial t} + \mathbf{u}_I \cdot \nabla\phi_G = 0$ to obtain $\tilde{\phi}_G^n$ from ϕ_G^{n-1}}
2: **compute bulk shape as a union** $\{\phi_B = \min(\tilde{\phi}_G^n, \phi_L)\}$
3: **redistance bulk shape** {Solve $|\nabla\phi_B| = 1$ on $\Omega_B^{[0;\alpha_M]}$}
4: ϕ_G = **ComputeGuide(**ϕ_B**,**ϕ_S**,**α_M**,**α_b**,**α_a**,**α_d**,**α_e**)**
5: **intersect** ϕ_G **with preliminary guide** $\{\phi_G = \max(\phi_P, \phi_G)\}$
6: **redistance** ϕ_G {Solve $|\nabla\phi_G| = 1$ on $\Omega_G^{[-\alpha_b;\alpha_b]}$}

The sequence of operations in Algorithm 2, notably the redistancing operations, may be expensive when computed at the same resolution as the guided liquid simulation. However we have found Algorithm 2 can be run at lower resolution than the guided liquid simulation without any apparent issues. This explains why the guided simulation times are lower than the timings for a full unguided liquid simulation at the same resolution (Table 2). For input shapes originating from simulations we perform the computations of the guide shapes at the resolution of the input simulation.

3.2.3 Computing the Reseeding Volume

The reseeding volume is computed immediately after the final guide shape, with the goal of filling gaps which open up between the guide shape and the guided liquid. In a particle based liquid simulator (such as Naiad's FLIP mode), liquid particles are reseeded inside the reseeding volume; in a level set based liquid simulator, a union is formed of the reseeding volume and the guided liquid. In both cases, the liquid volume is instantiated with the guiding velocities available either from the input data or computed via one of the algorithms described in section 3.3. Note that this is *not* the velocity by which the guide shape evolves. Fluid particles that move into the guide volume are removed.

The overall idea of the reseeding volume computation is to subtract the guide shape at the current time step ϕ_G^n from the guide shape at the previous time step advected forward to the current time step $\tilde{\phi}_G^n$. In particular gaps between $\tilde{\phi}_G^n$ and the guided liquid should appear for a single time step only due to numerical error. An outline is given by Algorithm 3. In our implementation we avoid reseeding particles closer than a certain threshold to the air-liquid interface to avoid causing noise at the liquid surface. To ensure the reseeding does not miss holes at the scale of a single grid cell we compute the reseeding volume at the resolution of the simulation, as opposed to the guide shapes that are computed at lower resolution. Our typical values for the parameters are $\alpha_s = 2.5\Delta x$ and $\alpha_t = \Delta x$.

Algorithm 3 ϕ_R = computeReseedingVolume

Input: $\tilde{\phi}_G^n$ {guide at previous time step advected forward}
Input: ϕ_G {guide at current time step}
Input: α_b {final narrow band radius of ϕ_R}
Input: α_s {safe region overlap between $\tilde{\phi}_G^n$ and ϕ_L}
Input: α_t {safe region overlap between $\tilde{\phi}_G^n$ and ϕ_G^n}
Require: $\{\alpha_s, \alpha_t\} < \alpha_b$
1: **dilate advected guide** $\{\phi_R = \tilde{\phi}_G^n - \alpha_s\}$
2: **erode and subtract current guide** $\{\phi_R = \max(\phi_R, -(\phi_G + \alpha_t))\}$
3: **redistance** {Solve $|\nabla\phi_R| = 1$ on $\Omega_R^{[-\alpha_b;\alpha_b]}$}

3.3 Velocity Interpolation

Input shapes originating from animations typically only have velocities defined at the surface as opposed to throughout their interior. However, we require interior velocities both for setting the velocity boundary condition and for reseeding. Note that the apparent velocity of the evolving guide shape is physically meaningless, due to construction via level set operations, and cannot be used—arbitrarily high velocities can appear when the fluid thins to less than the guide depth, for example. In this section we describe two methods for interpolating the velocity from the surface into the interior of the input shape (with a comparison given in section 4): extrapolating velocity in the normal direction from the input surface, and solving for the potential flow which matches the normal component of velocity at the input surface. Velocity interpolation is performed as part of the precomputation stage of the preliminary guide shape described in section 3.2.1.

3.3.1 Velocity Extrapolation in the Normal Direction

By default we propagate velocities from the surface into the interior of the input shape by constant extrapolation in the normal direction. Our solver does this with a discrete closest point transform: every unknown velocity value on the grid is set equal to the closest known sample. The produced velocity field is usually not divergence-free

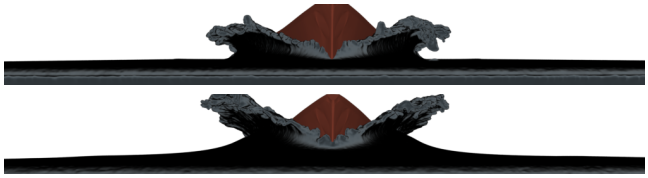


Figure 5: *Simulation of a moving boat. A certain simulation depth is required to obtain large-scale wave disturbances. Top row is the unguided high resolution shallow depth simulation. Bottom row is the guided high resolution simulation.*

but this approach is generally faster than computing the potential flow solution described next.

3.3.2 Potential Flow

Potential flow models incompressible inviscid irrotational flow. As opposed to simple extrapolation it gives a divergence-free velocity field, provided the surface velocities integrate to a flux of zero.

We solve for a scalar potential ψ whose gradient will be the interpolated velocity $\mathbf{u} = \nabla\psi$. Incompressibility gives the Laplace equation $\nabla \cdot \nabla\psi = 0$ in the volume, with a Neumann boundary condition $\partial\psi/\partial\hat{\mathbf{n}} = \mathbf{u}_I \cdot \hat{\mathbf{n}}$ to match the normal component of the input velocity at the surface of the input geometry [Lautrup 2005].

We discretize the continuous formulation on a staggered grid using central differences and the velocity potential ψ defined at cell-centers. The resulting linear system is solved independently on each connected component of the input shape. We refer to Briggs *et al.* [2000] for a detailed description of how to solve this type of equation system with pure Neumann boundary conditions.

4 Results and Discussion

We have integrated the methods proposed in this paper with Naiad, a FLIP-based liquid solver [Zhu and Bridson 2005] that uses a sparse volumetric data structure for storing auxiliary level sets and velocities. Voxels are defined in small tiles close to the surface and in the interior of the liquid, but not constrained to any particular rectangular domain. All dimensions are reported in world space units (metres) and in addition we report the maximum simulation domains in voxels. Timings as well as particle and grid cell counts for all examples in the paper are listed in table 2.

Figure 1a shows a simulation with grid spacing $1.20m$ of a boat pulled up through a water volume with initial dimensions $241m \times 46.1m \times 231m$. The guide shape is computed at the resolution of the original simulation (grid spacing $1.20m$). The guided high resolution simulation in Figure 1c has a grid spacing of $0.60m$, uses a minimum simulation depth of $6.0m$, has a maximum computational domain of $414 \times 225 \times 405$ voxels and has a computation time per time step of approximately $95s$. The unguided simulation in Figure 1d also has grid spacing $0.60m$, a water volume with shallower initial dimensions $241m \times 10.8m \times 231m$, a maximum computational domain of $441x \times 234 \times 423$ voxels and uses approximately $149s$ per time step, making it 50% slower yet without capturing the desired dynamics. An attempt to run the unguided simulation at high resolution and full depth caused the solver to run out of memory.

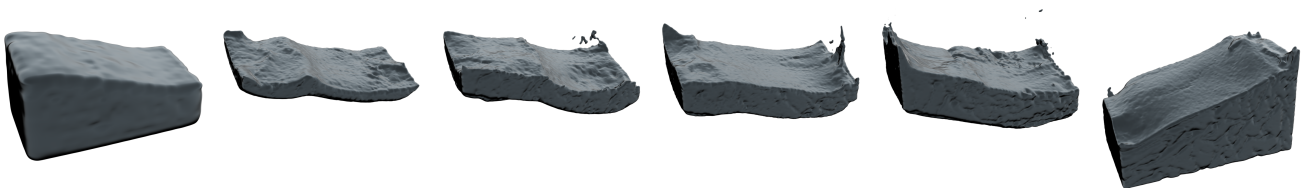
The example shown in Figure 6 illustrates that increasing the resolution alone can significantly change the timing of waves in a simulation. In particular Figure 6a and 6f depict the same simulation with grid spacings $0.04m$ and $0.01m$ respectively—note the difference in slope between the two surfaces. Figure 6b-e show simulations with varying simulation depths guided by the low resolution simulation in Figure 6a. Table 2 lists timings and storage requirements for the simulations. The video shows that the guided simulation largely follows the large-scale shape of the low resolution simulation. The difference between the low resolution and unguided high resolution simulation starts to become apparent about 80 frames into the animation. Occasionally the high resolution simulation will align with the low resolution simulation but will then quickly deviate again. The guided simulation in this example exhibits a tendency to create more high thin splashes than the low resolution guiding simulation; if undesirable this could be improved by combining our method with additional control away from the surface of the guiding shape. As expected, the effect of the guiding decreases as the depth of the guided liquid increases.

method	total time (seconds)	guide time (seconds)	number particles (millions)	number grid cells (millions)
Emerging boat, Figure 1				
guided	93.9	23.1	10.3	2.63
unguided shallow	149.0	-	19.7	4.80
Liquid in container, Figure 6				
guided, $\alpha_e = 0.15m$	6.44	.708	.623	.439
guided, $\alpha_e = 0.20m$	8.66	.701	.855	.490
guided, $\alpha_e = 0.25m$	9.93	.700	1.01	.530
guided, $\alpha_e = 0.30m$	10.7	.733	1.13	.553
unguided	13.9	-	1.67	.659
Sailing boat, Figure 7				
guided	91.1	9.24	12.0	4.65
unguided shallow	70.0	-	10.6	4.41
unguided deep	649.0	-	70.3	12.3
Water slide, Figure 9				
guided ($t = 1s$)	28.2	9.91	0.892	2.62
unguided ($t = 1s$)	35.3	-	2.34	3.82
guided ($t = 90s$)	209.0	31.8	7.83	22.8
unguided ($t = 90s$)	482.0	-	49.0	29.5

Table 2: *Timings on a machine with two Intel Xeon quad core 2.66GHz CPUs and 16GB of memory. The total time includes all steps of the solver, including loading (saving) data from (to) disk, for a single time step. The number of grid cells includes all grid cells allocated in voxel-tiles.*

Figure 7 illustrates a moving boat. The guided high resolution simulation retains the features of the guiding simulation with deeper water such as a relatively high wake and large-scale wave disturbances (Figure 5). The unguided high resolution simulation at a shallow simulation depth fails to capture these features. For this type of guided simulation where the guiding shape exhibits large tangential movements (in particular the waves in front of the boat), we observed that the depth of the guided liquid has to be relatively shallow in order to be guided properly. The guided high resolution simulation takes approximately $91.1s$ per time step, the shallow unguided high resolution simulation takes approximately $70s$ per time step and the unguided simulation at high resolution and full depth takes approximately $649s$ per time step (hence we obtain a $\times 6$ speedup).

Figure 9 shows the results of a simulation guided by an animated input shape, and Figure 8 compares the solution using guiding velocities computed from velocity extrapolation and potential flow. For this simulation the computation time per time step varied between $35s$ and $482s$ for the unguided simulation, and between $28s$ and $209s$ for the guided simulation. The solution using velocity extrapolation appears to have a tendency to travel faster than the



(a) (b) (c) (d) (e) (f)

Figure 6: *Frame 200 from a simulation of liquid moving back and forth inside a container.* ***(a)*** *Unguided simulation with grid spacing 0.04m.* ***(b-e)*** *Guided simulations with grid spacing* $0.01m$ *and simulation depths* $0.15m$, $0.20m$, $0.25m$, $0.30m$ *respectively.* ***(f)*** *Unguided simulation with grid spacing* $0.01m$*. At the starting configuration the bounding box is* $1.40m \times 1.42m \times 1.99m$*, and the maximum simulation domain at high resolution is* $162 \times 152 \times 216$ *voxels.*

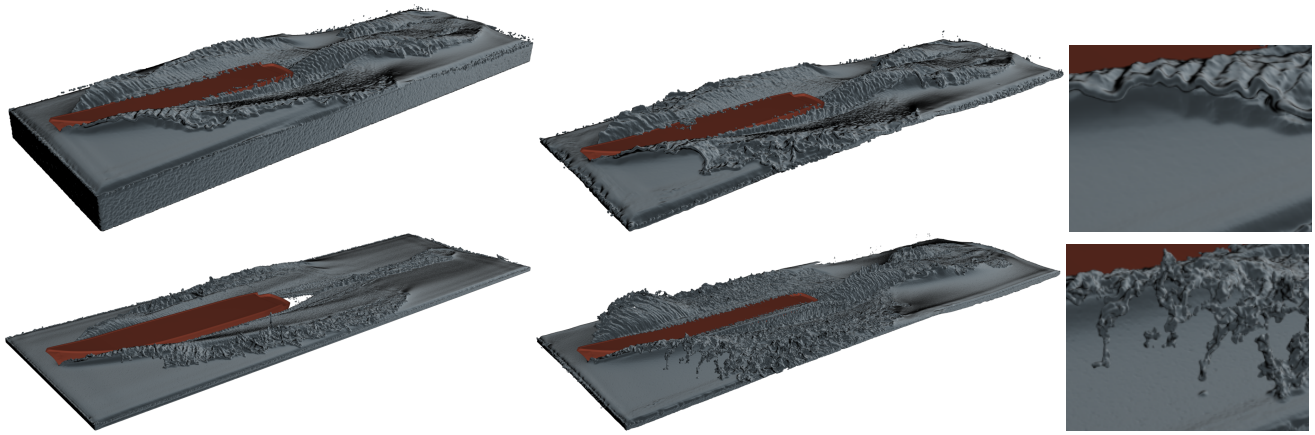


Figure 7: *Frame 164 from a scene with a moving boat. Left column depicts unguided simulations, middle column depicts guided simulations and the right column shows the gain in detail for a guided simulation obtained by increasing the resolution. The top (bottom) row shows simulations with grid spacing* $1.20m$ $(0.60m)$*. The unguided simulation in the top-left corner is used to guide the guided simulations that have a minimum simulation depth of* $4.5m$ *and a maximum simulation domain of* $306 \times 117 \times 837$ *voxels at high resolution. The initial dimensions of the water volume for the unguided simulations are* $158m \times 28m \times 476m$ *(top-left) and* $158m \times 5.0m \times 476m$ *(bottom-left), and the maximum simulation domain at high resolution is* $306 \times 72 \times 837$ *voxels.*

(a) (b) (c)

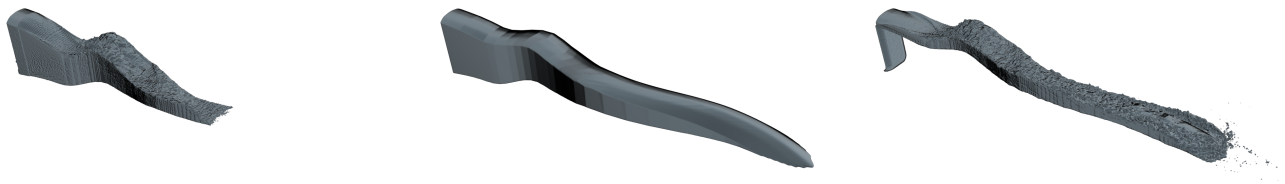
Figure 8: *Frame 90 from water guided by an animated input shape.* ***(a)*** *Simulation guided by extrapolated velocities.* ***(b)*** *The input shape.* ***(c)*** *Simulation guided by potential flow velocities.*

input shape, which could be problematic in cases where the timing is important. A comparison of velocity extrapolation and potential flow for the simulation in Figure 6 is available in the accompanying video and shows that velocity extrapolation also has a tendency to exaggerate splashing behaviour. Velocity extrapolation is generally faster than potential flow, but since this is done as a precomputation stage it does not affect the actual simulation time.

Since our guiding method imposes guiding only at the guide shape, input shapes moving with speeds that are far from physical may cause liquid to move inconsistently with the guide shape. Hence it would make sense to investigate combining our method with additional guiding inside the simulated liquid such as [Thürey et al. 2006] or [Nielsen et al. 2009]. Additionally, an animated guide does not necessarily take obstacles into account which may cause discrepancies when the guided liquid interacts with them. Note also that since we are simulating a relatively thin shell of liquid, we cannot capture guiding effects below this depth. Furthermore, our method cannot be used to guide thin splashes in the high resolution guided simulations, unless the splashes are to some extent captured at the resolution of the guiding simulation.

5 Conclusion and Future Work

We coupled a velocity boundary condition with velocity interpolation and a novel algorithm for computing guide shapes to achieve art-directed yet naturalistic liquid simulations at high resolution. We demonstrated that our method improves the correspondence between low and high resolution simulations, can decrease computation time for simulations that require relatively deep water and be used with animated input shapes. As future work it would be interesting to make a comparison to the divergence free velocity extrapolation method proposed by Rasmussen *et al.* [2004]. Currently we are also exploring the possibility of combining our method with other ways of controlling the liquid to allow for deployment in a wider range of scenarios. Our method is applicable in a visual effects production environment, and we hope that our work may serve as a starting point for further research into art direction of liquids.



(a) (b) (c)

Figure 9: *Frame 69 from water guided by an animated input. The simulations have a grid spacing of* $0.025m$ *and guide shapes* $0.05m$*.* ***(a)*** *Unguided simulation with liquid constantly emitted with the velocity of the input shape at the entry point to the simulation domain. Maximum simulation domain is* $8.55m \times 7.65m \times 30.15m$ *(*$342 \times 305 \times 1205$ *voxels).* ***(b)*** *The animated input shape.* ***(c)*** *The guided simulation with a minimum simulation depth of* $0.30m$ *and a maximum simulation domain of* $9.45m \times 13.7m \times 32.4m$ *(*$377 \times 522 \times 1295$ *voxels).*

Acknowledgements

We thank the anonymous reviewers for their feedback, Joe Letteri and Sebastian Sylwan for supporting our work, all members of the Weta Research Group as well as Brian Goodwin, Dave Gouge, Simon Le Grand, Marcus Nordenstam, Lucas Putnam, Kevin Romond, Christoph Sprenger, Jonathan Swartz, Diego Trazzi and Natasha Turner for their help.

References

ANGST, R., THÜREY, N., BOTSCH, M., AND GROSS, M. 2008. Robust and Efficient Wave Simulations on Deforming Meshes. *Computer Graphics Forum 27 (7)* (October), 6, 1895 – 1900.

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

BRIGGS, W. L., HENSON, V. E., AND MCCORMICK, S. F. 2000. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. In *ACM SIGGRAPH 2004 Papers*, 441–448.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *CGI '97: Proc. 1997 Conf. on Computer Graphics International*, IEEE Computer Society, 178–188.

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

HORVATH, C., AND GEIGER, W. 2009. Directable, high-resolution simulation of fire on the gpu. In *ACM SIGGRAPH 2009 Papers*, 41:1–41:8.

IRVING, G., GUENDELMAN, E., LOSASSO, F., AND FEDKIW, R. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *ACM SIGGRAPH 2006 Papers*, 805–811.

KIM, T., THÜREY, N., JAMES, D., AND GROSS, M. 2008. Wavelet turbulence for fluid simulation. In *ACM SIGGRAPH 2008 papers*, 50:1–50:6.

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

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM SIGGRAPH 2004 Papers*, 457–462.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 449–456.

MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2004. Animation and control of breaking waves. In *Proc. ACM/Eurographics Symp. Comp. Anim.*, 315–324.

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 resolution. In *Proc. ACM/Eurographics Symp. Comp. Anim.*, 206–215.

PATEL, S., TESSENDORF, J., AND MOLEMAKER, J. 2009. Monocoupled 3D and 2D river simulations. In *Proc. ACM/Eurographics Symp. Comp. Anim., Posters Session*.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D. Q., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. P. 2004. Directable photorealistic liquids. In *Proc. ACM/Eurographics Symp. Comp. Anim.*, 193–202.

SACHS, I., TWIGG, C. D., UREN, L., PEARSON, D., AND RASMUSSEN, N. 2010. Waterbending: Water effects on "The Last Airbender". In *ACM SIGGRAPH 2010 Talks*.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. In *ACM SIGGRAPH 2005 Papers*, 910–914.

SETHIAN, J. A. 1996. A fast marching level set method for monotonically advancing fronts. *Proc. of the National Academy of Sciences of the USA 93*, 4 (February), 1591–1595.

SHI, L., AND YU, Y. 2005. Controllable smoke animation with guiding objects. *ACM Trans. Graph. 24*, 1, 140–164.

SHI, L., AND YU, Y. 2005. Taming liquids for rapidly changing targets. In *Proc. ACM/Eurographics Symp. Comp. Anim.*, 229–236.

THÜREY, N., KEISER, R., PAULY, M., AND RÜDE, U. 2006. Detail-preserving fluid control. In *Proc. ACM/Eurographics Symp. Comp. Anim.*, 7–12.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. In *ACM SIGGRAPH 2003 Papers*, 716–723.

TROJANSKY, S. 2008. Raging waters: the rivergod of Narnia. In *ACM SIGGRAPH 2008 Talks*, 74:1–74:1.

WIEBE, M., AND HOUSTON, B. 2004. The tar monster: creating a character with fluid simulation. In *ACM SIGGRAPH 2004 Sketches*, 64.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, 965–972.