Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
P. Kry and J. Lee (Editors)

# Controlling Liquids Using Meshes

Karthik Raveendran[1] Nils Thuerey[2] Chris Wojtan[3] Greg Turk[1]

[1]Georgia Tech
[2]Scanline VFX
[3]IST Austria

**Abstract**

*We present an approach for artist-directed animation of liquids using multiple levels of control over the simulation, ranging from the overall tracking of desired shapes to highly detailed secondary effects such as dripping streams, separating sheets of fluid, surface waves and ripples. The first portion of our technique is a volume preserving morph that allows the animator to produce a plausible fluid-like motion from a sparse set of control meshes. By rasterizing the resulting control meshes onto the simulation grid, the mesh velocities act as boundary conditions during the projection step of the fluid simulation. We can then blend this motion together with uncontrolled fluid velocities to achieve a more relaxed control over the fluid that captures natural inertial effects. Our method can produce highly detailed liquid surfaces with control over sub-grid details by using a mesh-based surface tracker on top of a coarse grid-based fluid simulation. We can create ripples and waves on the fluid surface attracting the surface mesh to the control mesh with spring-like forces and also by running a wave simulation over the surface mesh. Our video results demonstrate how our control scheme can be used to create animated characters and shapes that are made of water.*

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation Simulation and Modeling [I.6.8]: Types of Simulation—Animation.

## 1. Introduction

Over the last decade, advances in fluid simulation and rendering have helped animators synthesize photorealistic shots for movies that would have been virtually impossible to create by hand. Current techniques can efficiently simulate a wide gamut of fluid behavior, ranging from small scale effects such as swirling and splashing of a glass of water, to massive natural phenomena such as flooding rivers and crashing ocean waves. Despite the advent of these computational methods, fluid simulation in movie production still involves a large degree of trial and error. One of the main reasons for this is the highly non-linear and dynamic nature of water, which can cause it to quickly deviate from the initial conditions prescribed by the animator.

This task is further complicated by more artistic design goals such as creatures or shapes made of water (as seen in numerous movies like the Chronicles of Narnia: Prince Caspian or Avatar: The Last Airbender). In these cases particularly, it is not adequate to merely force water to form these shapes; we also desire the dynamic effects that we typically associate with water such as splashes, dripping streams, surface waves and ripples. Certain existing

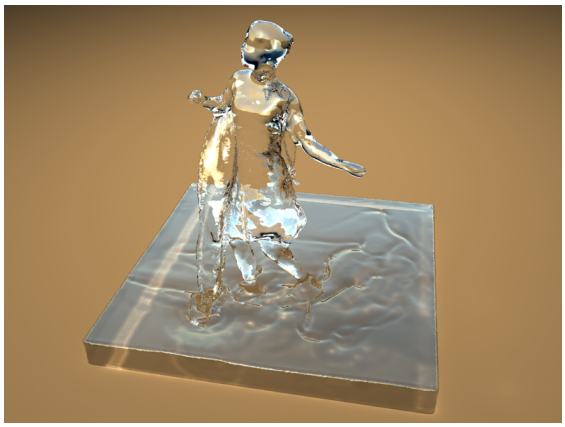

**Figure 1:** *A dancer made of water.*

techniques for fluid control can successfully track desired shapes, but either tend to provide minimal control over secondary motions or sacrifice them altogether. On the other side of the coin, recent methods [NCZ*09, NB11, HMK11] have aimed to loosely guide the output of higher resolution simulations to that of lower resolution versions which are frequently used in pre-visualization. These have primarily been intended for use in the simulation of realistic scenarios.

In this paper, we focus on a stronger form of control with an emphasis on the liquid surface and favor artistic goals over more naturalistic phenomena. We propose a new method for creating high quality fluid animations that provides the animator with multiple levels of control over the simulation while meeting the basic design goals for the bulk motion of the liquid. Our method takes in a dense sequence of *control meshes* as its input. This animated mesh sequence can either be directly provided by the animator or can be generated from a sparse set of user-defined input meshes using our volume-preserving morphing technique. The animator can then run the simulator and adjust some simple dials that select the *drippiness* or *looseness* of the water surrounding the target shape. In addition, the animator can add specific surface details such as bumps or introduce surface waves to create a more dynamic look while still matching the motion prescribed by the given control meshes.

The key contributions of our paper are as follows:

- Volume-preserving morphing: We introduce an optimization framework that produces a dense sequence of control meshes from a sparse set of user defined meshes. Our morphs preserve volume and support positional and velocity constraints.
- Eulerian control: We compute control forces on the grid using the velocity of the control meshes as boundary conditions on the pressure solve. This allows us to not only reproduce the bulk flow from the keyframes, but also allow for natural inertial effects such as dripping or separating sheets of water from a fast moving shape.
- Mesh-based details: We can generate highly detailed water surfaces and effects (such as ripples and gravity waves) by using a mesh-based representation of the surface that is attracted to the user-provided control meshes.

Our proposed scheme clearly divides control over the basic underlying motion from that of secondary effects of the fluid and this makes it intuitive for the animator to fine-tune specific aspects of the animation. Unlike previous approaches to this problem, we can control details of the liquid surface at scales smaller than a single grid cell. This freedom allows us to obtain high fidelity effects at reasonable grid resolutions.

## 2. Related Work

Foster and Metaxas [FM97] introduced the notion of fluid control to computer graphics through the use of embedded controllers. Since then, a number of researchers have addressed the challenge of making fluids closely track a set of user-defined target shapes. [FL04], [HK04] and [SY05a] proposed methods that control smoke to form and follow moving target shapes. [SY05b] introduced a new technique for liquids with free surfaces where a proportional derivative (PD) controller was used to track rapidly changing targets. Our work shares certain similarities with these control methods in that it also uses Neumann boundary conditions on an Eulerian grid as a means of applying control forces. However, due to the construction of our boundary conditions, we can achieve close tracking without the need for a carefully tuned PD-controller.

[REN*04] and [TKPR06] take a different approach to this problem by using control particles to adjust velocities, viscosity and other properties of the liquid. These techniques avoid aliasing artifacts on the grid by using this form of Lagrangian control. Higher level objectives are harder to achieve because these methods require the artist to translate the desired visual behaviors into specific physical quantities. Instead of using a particle levelset as in [REN*04], we use a Lagrangian surface tracking technique with a coarse Eulerian simulation to handle fine details.

[MTPS04] proposed a novel method of dealing with keyframed control of fluids. They treat a fluid simulation as a composition of functions and minimize an objective function to solve for the control forces. For reasons of efficiency, they applied the adjoint method to compute derivatives from the simulation with respect to each control parameter. However, this technique still requires the calculation and storage of these derivatives for the entire simulation at each timestep and as a consequence, it is computationally expensive. This makes it impractical for use at high grid resolutions that are necessary for capturing details.

A more recent class of methods ( [NCZ*09, NB11, HMK11]) has aimed to make higher resolution simulations loosely match the behavior of their lower resolution counterparts with the same initial conditions. These are intended for creating production quality output while allowing the artists to prototype using faster, low resolution simulations and are targeted at naturalistic scenarios. Our work shares certain commonalities with [NB11] in terms of the potential flow solution that used to compute interior velocities, however our focus is to generate more supernatural phenomena such as creatures made of water and in particular, on highly detailed yet controllable liquid surfaces.

Our fluid simulator draws upon some of the tools that are commonly used for animating fluids in graphics. We perform fluid simulation using an Eulerian framework on a staggered MAC grid, as was introduced to graphics by [FM96]. We enforce fluid incompressibility by solving a Poisson equation, as in [Sta99]. Details of these techniques can be found in [Bri08]. For fine surface details, we make use of a mesh-based surface tracker, as in [WTGT10], and we carry out wave dynamics on surfaces using the methods of [TWGT10].

Researchers have proposed various solutions to the prob-

lem of volume-preserving deformation such as [HML99, AB97, vFTS06]. Similarly, morphing is a well studied problem in computer graphics (see [Ale02] for a survey of recent approaches). However, to our knowledge, there has been no work that specifically addresses the issue of volume-preserving morphing. It is important to recognize that *deformation* and *morphing* (and hence volume-preserving variants of these) are not the same. Deformation is like an initial value problem, where an initial shape is modified by the user's guidance. Morphing is more like a boundary value problem, in which both the initial shape and the target shape are exactly specified by the user. Except in extremely simple cases, it is not possible for an artist to use deformation tools alone to reach a specific target shape. In this work, we present a new algorithm for efficiently morphing between two shapes while preserving volume. We use this to generate keyframes for fluid animations.

## 3. Algorithm Overview

The input to our fluid control system is a set of per-frame *control meshes* that are provided by the animator. These could either be generated from a sparser set of triangle meshes using the morphing algorithm described in Section 4 or could be provided independently by the animator. The only condition imposed on these control meshes is that they must maintain vertex correspondences and topology.

In order to obtain secondary motions, we incorporate these control meshes into a fluid simulator. To compute the Eulerian control forces, the surface of the control mesh for each timestep is rasterized onto the simulation grid and its velocities (computed using finite differences between successive frames) are transferred onto the cell faces of the fluid grid. By setting these velocities as boundary conditions in the Poisson equation, we can closely track our desired shape or blend it with an unforced velocity field to achieve a looser form of control. We describe the details of this in Section 5.

To add surface details, we use a coarse grid-based fluid simulation that is coupled with a Lagrangian surface tracker (as described in [WTGT10]). This surface tracker makes use of a *surface mesh* in order to represent the fluid surface. Control forces are computed on both the grid and the surface mesh and this helps us achieve much higher detail than previous methods without incurring a significant computational overhead. The pressure projection step from Section 5 produces a divergence-free velocity field. We advect the surface mesh through this divergence-free field and handle all of the topological changes needed to maintain the surface mesh. Finally we compute control forces on the surface mesh vertices to produce a variety of effects ranging from tracking of sub-grid details to the generation of surface waves (see Section 6)

## 4. Volume-Preserving Morphing

Morphing is a effective way of producing an animation from a sparse set of user defined control shapes. In a typical use case, the artist provides a set of triangulated meshes (say poses for a character or deformations of a 3D model) at different instants of time, as well as correspondences between vertices. The morpher then performs some form of interpolation between each pair of corresponding vertices and produces a set of in-between shapes. If these shapes were made of water, then they must preserve their volume through the morph since water is incompressible. However, most commonly used morphing techniques do not guarantee this property. In this section, we describe a novel technique for performing volume-conserving morphs between shapes.

The input to our morpher is a set of control meshes $K$ defined by closed, manifold, connected triangulated meshes at user specified instants of time $t$: $K = \{K_t\}$. We require that vertex correspondences and connectivity of the mesh are preserved between these shapes. The user also needs to specify the number of in-between frames $T$ to be generated between each pair of given meshes. Since the connectivity of the mesh is assumed to be fixed, each in-between frame $M_t$ is fully defined by the positions of the $V$ vertices of the mesh, $x_i^t$ where $t$ is the instant of time and $i$ is the vertex number.

The goal of the morphing algorithm is to produce a motion that is as smooth as possible while conserving volume. This amounts to minimizing the following objective function:

$$F = \sum_{t=1}^{T} \sum_{i=1}^{V} \left\| v_i^{t+1} - v_i^t \right\|^2 \qquad (1)$$

$$\text{where } v_i^t = \frac{x_i^{t+1} - x_i^t}{\Delta t}$$

subject to the volume constraint:

$$\text{Volume}\,(M_t) = \text{constant}, \forall t \in T \qquad (2)$$

By taking the derivative of Equation 2 with respect to time and by applying the Divergence theorem, we obtain a transformed constraint that is purely defined using quantities on the surface of the mesh:

$$\sum_{i=1}^{F} v_{f_i}^t \cdot n_i^t S_i = 0, \forall t \in T \qquad (3)$$

$$\text{where } v_{f_i}^t = \frac{\sum_{j=1}^{3} v_j^t}{3}$$

Here, $v_{f_i}^t$, $n_i^t$ and $S_i$ denote the velocity, normal and area of the triangular face $f_i$ respectively. We compute velocities using forward differences between vertex positions. Unfortunately, the divergence constraint in Equation 3 is nonlinear because $n_i^t$ and $S_i$ depend on the vertices of the triangle. Further, our optimization is fairly high dimensional with $T \times 3V$ variables and this makes it extremely difficult to obtain a global minimizer.

In order to make this computation tractable, we linearize

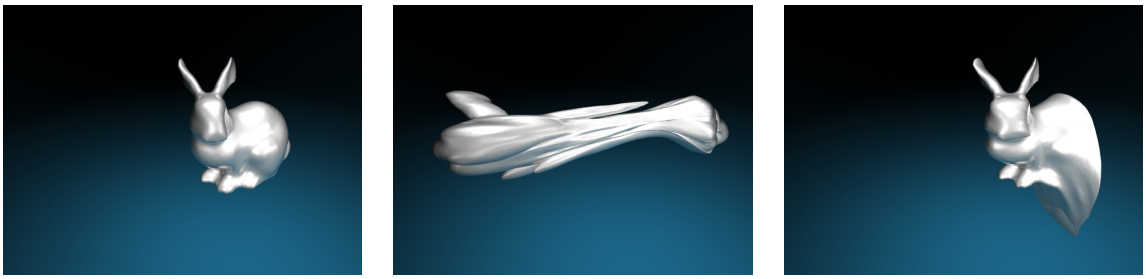

**Figure 2:** *Input meshes to our volume-preserving morpher. The final simulation sequence that uses the morpher output is shown in Figure 6.*

Equation 3 by approximating the normal and area at time $t$ by interpolating between the known quantities at the the start and end control meshes defined by the user. We use a simple linear interpolation for the area of the triangle, but for the normal, we perform a spherical interpolation between the quaternion representations of the start and end normals. This makes the equation linear in velocities, and consequently, in vertex positions. In practice, this approximation for the normals turns out to be a good starting point and can be adjusted over multiple iterations.

We can introduce a Lagrange multiplier $\lambda_t$ per constraint, and obtain a new objective function that needs to be minimized with respect to $x_i^t$ and maximized with respect to $\lambda_t$:

$$F_{new} = \sum_{t=1}^{T}\sum_{i=1}^{V} \|v_i^{t+1} - v_i^t\|^2 + \sum_{t=1}^{T} \lambda_t \left( \sum_{i=1}^{F} {v_f}_i^t \cdot n_i^t S_i \right) \quad (4)$$

By taking the partial derivatives with respect to each free variable and setting them to zero, we obtain a system of linear equations in $x_i^t$ and $\lambda_t$, with a total of $T \times (3V+1)$ variables. This results in a sparse symmetric indefinite linear system (Equation 5) that is well studied in linear algebra.

$$\begin{bmatrix} A & B \\ B^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \quad (5)$$

There are many ways to solve this linear system such as LU or LDL factorizations of the matrix ( [GVL96]), null space methods that eliminate the constraints ( [GHN01]) or approaches that rely on the range space ( [GMSW87]). We use a range-space method in this paper by computing the Schur complement of the matrix. The Schur complement transforms the problem of solving a $(m+n) \times (m+n)$ system into one that requires inverting two smaller matrices ($m \times m$ and $n \times n$). This is particularly useful when one of the smaller matrices has some property (such as sparsity or symmetric positive definiteness) that permits the fast computation of a inverse.

The solution to our linear system can be computed as follows:

$$\begin{bmatrix} A & B \\ B^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \quad (6)$$

$$\lambda = \left(B^T A^{-1} B\right)^{-1} \left(B^T A^{-1} \mathbf{p} - \mathbf{q}\right) \quad (7)$$

$$\mathbf{x} = A^{-1}\left(\mathbf{p} - B\lambda\right) \quad (8)$$

We need to compute the inverse of the matrix A in order to proceed with the Schur complement solve. If the elements of the vector $\mathbf{x}$ are ordered sequentially in time, we obtain a symmetric five-band stencil due to the forward difference discretization. This causes A to be sparse, symmetric and positive definite. We compute the Cholesky decomposition of A using the Cholesky-Banachiwicz algorithm. This yields a lower triangular matrix L that has the same sparsity and structure as the matrix A and can be computed in $O(n)$ instead of the typical $O(n^2)$ since there are exactly three non-zero elements per row in the decomposition. We further observe that L is block diagonal and this can be exploited to compute its inverse ($L_{inv}$) efficiently by inverting all unique $T \times T$ blocks of L using a LU decomposition and copying these to other identical blocks in the matrix.

It is interesting to note that A is completely determined by the number of vertices of the input shapes and the number of in-between frames. As a result, the inverse of A can be precomputed and reused for a variety of morph targets as long as the number of vertices is kept fixed. Once we have the inverse, we can compute the Lagrange multipliers that enforce the constraints and then solve a $T \times T$ system using LU decomposition to obtain our inbetween positions $\mathbf{x}$. Typically, the number of inbetween frames ($T$) ranges from 30 to 60 and this does not prove to be a computational bottleneck.

Our basic implementation (without using optimized BLAS libraries) can solve large systems (1,000,000 free variables) in less than 30 seconds without any numerical instabilities. In contrast, an iterative solver such as GMRES did not converge on a solution in a majority of our test cases, while direct LU decomposition became prohibitively expensive for such large matrices.

Note that the normals and triangle areas resulting from this solve may differ from our predicted versions used to construct the linear system. In such cases, we compute a

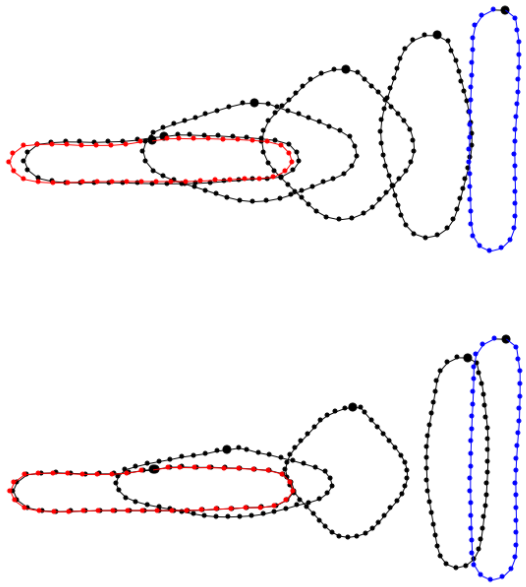

**Figure 3:** *Comparison of linear interpolation (top) with our volume preserving morph (bottom). There is a volume gain of more than 50% in the middle shape for the linear morph.*

new set of normals and areas from the previous solution and repeat the process until convergence. In practice, this converges using one or two iterations for our test cases including those involving large rotations. These iterations are extremely inexpensive because the matrix A does not depend on the normals or the triangle area and this allows us to reuse its inverse computed at the beginning of the solve. We can also incorporate additional positional or velocity constraints while maintaining volume by introducing more Lagrange multpliers.

We opted to constrain the total divergence at each frame to be equal to zero. In other words, we chose to constrain the derivative of the volume with respect to time, instead of constraining the volume directly. The main reason behind this is that the convergence of the iterative process is much faster with the weaker derivative constraint while a direct volume constraint produces significant oscillations. Our results did not improve when we used the even weaker, second derivative of volume as a constraint and instead, this caused a noticeable loss in volume.

Figure 3 shows a 2D scenario where our volume conserving morph manages to maintain the original volume while a simple linear interpolation causes an expansion of over 50%. Our morphing technique offers an easy way for animators to produce a smooth deformation between shapes while conserving volume without the use of a skeleton or a character rig. Further, it offers control over the shape by letting the user specify target velocities or positions for parts of the shape through the course of the morph. The biggest benefit is that we can use the resulting velocity field (from finite differences between corresponding vertices) directly inside a fluid simulator and track the animation perfectly since these velocities are not modified by the pressure projection step. This allows the animator to focus on creating the basic motion independent of the simulator. The bunny deformation sequence in Figure 2 shows the results of using our mesh morphing for a 3D animation, and Figure 6 shows using these control meshes together with fluid simulation. Note that it is indeed possible to use this formulation to morph between shapes that differ in volume by interpolating the quantity over the inbetween frames and accounting for the change on the right hand side of equation 3. In general, animations that mostly conserve volume tend to look more natural than those that do not.

## 5. Eulerian Control

The primary reason for using a simulation instead of procedurally animating the provided shapes is to obtain secondary motions that increase the naturalness of the scene. However, any algorithm for fluid control must at the very least, produce results that match the user specified shapes. This notion is not strictly restricted to the geometry of the output but also extends to other aspects of the motion such as the velocity and acceleration. For instance, we would expect to see fast moving water in regions of a rapidly deforming or translating target. This forms the *raison d'etre* for our grid-based control force.

Our Eulerian fluid simulator makes use of a staggered MAC grid, which we shall refer to as the "grid" from this point on. Our simulator broadly consists of the following stages: integration of external forces, advection, and finally, pressure projection. The last step projects the velocities into the closest divergence-free field and enforces incompressibility. We can then advect our current fluid surface through this velocity field to obtain a new surface. While it may be appealing to simply add control forces in the first step of the simulation, this approach can produce unpredictable results because the added forces might be significantly altered by the projection stage. Instead, we enforce our control during the projection step by specifying boundary velocities along the control mesh and solve the resulting Poisson equation to obtain a conforming divergence-free velocity field through the entire domain.

This formulation works because the elliptic Poisson equation has the useful property that the solution within an enclosed region is completely determined by the values specified along its boundary. Since our target shapes are meshes that maintain correspondences across frames, we can compute the velocities at the vertices of each control mesh using a simple forward difference. We set directly set these as Neumann boundary conditions and obtain new velocities everywhere in the domain that respect these control velocities. The sole restriction here is that the flux through this boundary (i.e. the net amount of fluid flowing through this surface) must be zero in order to preserve volume. If the meshes

change in volume, we account for this by modifying the divergence on the right hand side of the Poisson equation.

In terms of implementation, we need to rasterize velocities from the control mesh onto the face of the simulation grid. We first rasterize the mesh onto the grid and ensure that the rasterized shape is connected by closing any small air pockets that might have been formed due to a mismatch in resolutions between the control mesh and the grid. While the velocity field defined on the vertices of the input mesh may have zero divergence, the rasterized field may not obey this property and needs to be corrected. We do this by computing the divergence along the rasterized boundary and distributing it evenly amongst the cells on the surface. We use a standard discretization of the Poisson equation and assign these rasterized boundary velocities as Neumann conditions. We solve the linear system using the preconditioned conjugate gradient method [Bri08].

### 5.1. Relaxed Control

Using the technique outlined above, the motion of the fluid will track the user defined control meshes closely. However, it is often desirable to have a less constrained motion of water that retains more of its inertia or momentum. We call this *relaxed control*. To do this, we perform an additional pressure projection step without enforcing the boundary conditions from the rasterized control mesh. Let $F_u$ denote the force required to project out the divergent component from the intermediate velocity field $u^*$ without any boundary conditions imposed by the control shape. The normal pressure projection required to enforce the rasterized control velocities results in force $F_v$ . Note that each of these results in a divergence-free velocity field and by linearly blending these, we obtain a new velocity field that also has zero divergence (Equation 9).

$$u^* + (1-\alpha)F_u + \alpha F_v = (1-\alpha)u + \alpha v = u_{relaxed} \qquad (9)$$

These relaxed control forces may not track the shape perfectly, but they can produce plausible secondary effects such as sheets tearing off a fast moving arm or more subtle sloshing that are consequences of the artist prescribed motion. There exists a distinction between a typical feedback control scheme and our relaxed control method. In the former, animators tune the value of various gains in order to closely match the shape and velocities of the target and the ideal value for each of these is unknown and varies with time due to the dynamic nature of the simulation. However, in our method, the blending coefficient determines the balance between exact tracking and natural motion and this makes for a more intuitive dial for controlling the animation. We demonstrate results of this relaxed control in the dancing sequence of Figure 1 and in our video.

### 5.2. Volume Refilling

When the control is relaxed to a large degree, water tends to spill out from the control shapes and this typically leads to voids inside the target shape that may be undesirable. We detect the loss of volume by supersampling all grid cells inside the rasterized target shape and checking if they are empty. We then modify the divergence of all cells in the interior of the rasterized target to compensate for the lost volume during the pressure projection. This is similar to the approach described in [KLL*07] except that we can accurately compute the volume loss because we know the desired volume of the target. This leads to natural refilling of the shape without resorting to stiff spring-like forces.

## 6. Mesh-based Details

The control scheme described thus far operates solely on the grid despite using control meshes as input. If this scheme alone was used to produce controlled fluid effects, it would require a high resolution grid in order to produce detailed fluid surfaces. Recent work by [TWGT10, WTGT10] has shown the benefits of using a mesh-based Lagrangian surface tracking technique coupled with a relatively coarse Eulerian simulation and in particular, these methods can produce impressive sub-grid surface details and motions. In the context of fluid control, such a mesh-based surface representation is attractive because simulation on a low resolution grid is significantly cheaper (the asymptotic complexity of the Eulerian simulation being $O(n^4)$ where $n$ is the grid resolution along one dimension). Consequently, artists can perform multiple iterations to perfect the bulk motion which is imparted by the grid before fine-tuning surface details.

In this section, we describe how we make use of just such a mesh-based surface tracker to give us more fine details and more control of the fluid surface. We will refer to this new mesh as the *surface mesh*, which is advected by the grid velocities and represents the fine details of the fluid surface. This new mesh should not be confused with

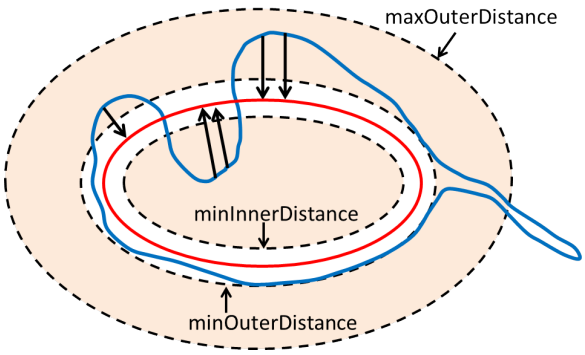


**Figure 4:** *Distance based cutoffs for the mesh attraction force. The user-supplied control mesh is red, and the surface mesh is in blue.*

the user-supplied *control meshes*. We will describe how this surface mesh allows us an additional layer of control above the previously described grid-based method to form fine details such as bumps, spikes and wrinkles, create surface waves and ripples and control the level of drippiness of water around the shape.

Prior to applying any control force on the surface mesh, we advect the surface through the velocity field produced by the grid based simulation. This velocity field includes the Eulerian control forces that help move the fluid surface towards the target. Next, we handle topological changes by performing operations on the mesh as described in [WTGT10].

### 6.1. Capturing Fine Details

The first form of mesh-based control aims to attract the current surface mesh towards the control mesh and match fine details that cannot be captured on the grid. For each vertex of the current surface mesh, we find the closest point on the control mesh. In our implementation, we use a kd-tree as a spatial data structure and insert all points of the control mesh into the tree. We can then query the tree to find the nearest point for a given point on the surface mesh. Next, we apply a positional update using a damped spring equation to pull the surface point closer to the control mesh. The user can adjust the strength of these springs to determine the speed at which the surface is attracted to the control mesh. However, a naive application of such a spring based force can suppress dynamics near the surface and also prevent the natural separation of water that is far away from the control mesh. We introduce tunable distance based parameters that limit the application of this mesh attraction force (see Figure 4). The control force only acts in the shaded regions and the transitions can be further smoothed by using a linear or quadratic falloff. Using this control technique, we can form sub-grid details such as wrinkles on a forehead, as can be seen in the right image of Figure 5.

### 6.2. Control Over Thin Sheets

One of the benefits of using a Lagrangian surface tracker is the preservation of thin sheets of water. We defined thin sheets as structures that are nearly flat and that are thinner than a grid cell. Thin sheets do not breakup naturally since they are unaffected by surface tension forces and can persist for long periods of time in the simulation until they are perturbed due to interaction with other fluid. These sheets can be used to impart a specific look to the animation. For instance, long trailing sheets can indicate high velocities and drippiness, and sheets that quickly break up can be used to preserve detail around important regions of the control mesh.

We can selectively decide to cause a thin sheet to tear, if this is desired for a given animation. To do this, we first identify vertices of the surface mesh that lie on thin sheets as those that lie in a thickened shell that is outside the range of the mesh attraction force and inside a grid cell that is not entirely surrounded by fluid (i.e. at least two of its neighbors along an axis must be air cells). Next, we apply several iterations of smoothing (using mean curvature flow) on only these vertices. This causes a natural tearing of the sheet starting at the minimum distance set by the artist (since vertices below this threshold are being pulled towards the target by the attraction forces). By altering the distance of application and the frequency of this smoothing step, these sheets can be detached at will. For instance, in the dancer animation, we run the thin sheet tearing step once in every five simulation steps. This allows for a more drippy look and creates highly detailed structures that tear off from the arms and torso and are flung away due to their large velocities.

### 6.3. Surface Waves

The previously described control helps the animator track specific details, but does not introduce any additional dynamics on the surface mesh. It is well known that surface waves such as gravity and capillary waves play an important role in creating realistic visual effects at small scales. Capillary waves are a result of surface tension forces that try to minimize the area of the liquid surface while retaining its volume. Due to this, these waves are unlikely to track our control meshes. This inspires us to introduce surface dynamics via waves that are compatible with the underlying animation given by the artist. In order to obtain well behaved waves on our liquid surface that can co-exist with our other control forces, we need to alter the standard linearized two-dimensional wave equation:

$$\frac{\partial^2 h}{\partial t^2} = c^2 \left( \frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} \right) \tag{10}$$

In this equation, $c$ defines the wave speed, while $h$ refers to the current height of the wave at that point on the surface. The right hand side is the curvature of the scalar field $h$, while the left hand side of the equation is the acceleration at this point which is proportional to this curvature. Intuitively, curvature can be viewed as a measure of the difference of the scalar value at the current point compared to the scalar values at neighboring points. If we set $h$ to be equal to the distance from a point on the surface mesh to the closest point on the control mesh, then the acceleration experienced by that point on the surface will be determined by its displacement from the control mesh relative to its neighbors. To illustrate the effects of this equation, consider a water surface that is parallel to the control mesh. The points on this surface will not experience any wave forces because they are equidistant from the control shape. Now, if the control mesh were to develop a slight furrow, this would change the relative distances and spawn a wave that will eventually settle down once the sheet takes the shape of the control surface.

Note that this sheet does not need to exactly settle down on the control mesh for it to not experience any force - a parallel displacement will also result in zero curvature. How-

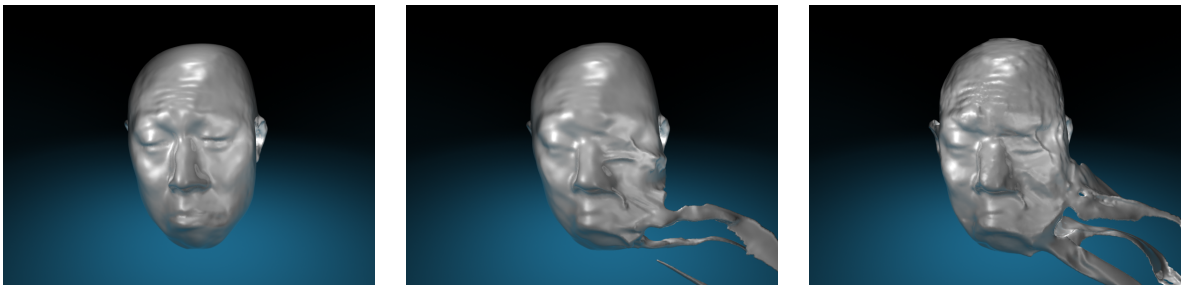

**Figure 5:** *Different effects can be achieved by varying the control parameters (from left to right): tight tracking, drippy with relaxed control, drippy with waves.*

ever, we can limit the magnitude of such a displacement by using the mesh attraction forces described in the previous section. Specifically, by tuning the *minInnerDistance* and *minOuterDistance* (see Figure 4), we can determine how far surface may move away from the control mesh before being pulled back. These vertices that move beyond the inner distance thresholds are damped and act as a damped boundary conditions in the wave equation solve. Further, if certain vertices have a high velocity component that is normal to the surface, they can quickly move beyond the influence of both wave and attraction forces. Hence, this setup does not preclude the possibility of liquids sheets detaching themselves from the main body of water and allows for waves to co-exist with our other control techniques.

We use the approach defined in [TWGT10] and solve this equation using the Implicit Newmark time integration scheme. This amounts to solve the following sparse linear system (where **h** is a vector of the heights of all surface points, and L is the matrix representation of the Laplacian operator on **h** ):

$$\left(I - \frac{\Delta t^2}{4} c^2 L\right) \mathbf{h}^{n+1} = \left(I + \frac{\Delta t^2}{4} c^2 L\right) \mathbf{h}^n + \Delta t \mathbf{h}_t^n \quad (11)$$

$$\mathbf{h}_t^n = 0.5\Delta t \left(\mathbf{h}_{tt}^{n-1} + \mathbf{h}_{tt}^n\right) \quad (12)$$

We update the vertex positions of the surface mesh by displacing them along their normals by the new height values obtained by solving the above equation. We do not include vertices that are beyond the inner distance thresholds while solving this equation.

In some cases it may be desirable to prevent waves in user defined regions to preserve features. We allow the user to specify vertices that must be preserved exactly by painting weights on the mesh. A weight of 1 implies that the vertex is unaffected by the wave equation and is pulled towards the control mesh using the mesh attraction force. A weight of 0 indicates that the point is moved solely by the wave equation. We then run the diffusion equation on the mesh for several iterations to spread these weights. The final displacement of a vertex is a linear blend (using this diffused weight) between the position predicted by the wave equation and the position determined by the attractive forces. This approximates non-reflecting boundary conditions reasonably and preserves details where desired. We used this approach to preserve the details of the mouth and eyes for the animation of Figure 5.

## 7. Results

Please watch the accompanying video to see the sequences that are described in this section. All of our fluid simulations were run on 8 cores of a dual processor Intel Xeon with 2.4 GHz CPU's and 6 GB of memory. Our control scheme typically adds an overhead of 15% - 30% to the computational cost of a regular fluid simulation. Most of the sequences averaged under 10 seconds per frame at a grid resolution of $100^3$. Note that the average edge length of a triangle in our surface mesh is typically less than one-third that of a grid cell. In some cases, this can lead to the preservation of certain sub-grid details such as creases. These may be removed by the application of the wave equation or smoothing on the surface.

The liquid bunny sequence of Figure 6 demonstrates our volume preserving morpher. In this sequence, a bunny is pulled out of a pool of water, it is deformed by twisting, stretching, and squashing, and then it is dropped back into the pool. For this sequence, we provided only a coarse sequence of control meshes for the various bunny shapes. Our volume preserving morpher then created the per-frame control meshes that were used for the sequence. The relaxed control parameter was set to $\alpha = 0.5$ for this scene in order to allow water to easily drip out of the deforming bunny. The pool of water in this example is not controlled at all, yet it interacts gracefully with the moving bunny.

In our video, we show a breakdown of some of the elements of the morphing bunny sequence. We first go through the morphs, without any fluid simulation. Note that we added velocity constraints to the stretch and squash sequences. This is especially important at the end of the squash sequence when we release control of the fluid and the momentum of the fluid creates a large splash. We also show examples of using different $\alpha$ (blending) values with the bunny, showing various degrees of relaxed control.

Figure 1 shows our water dancer example. The control

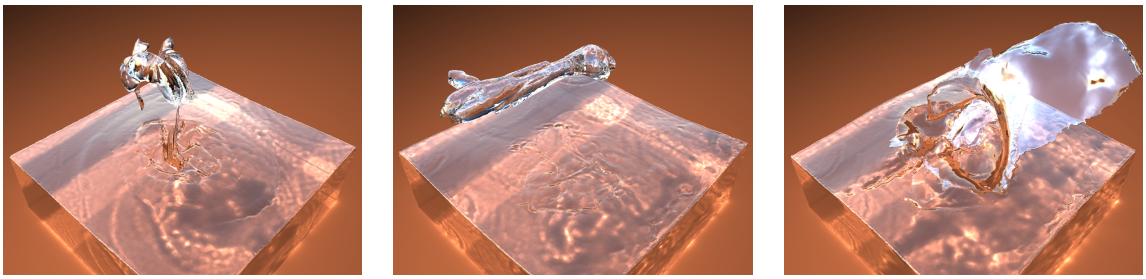

**Figure 6:** *Water bending: A bunny is pulled out of a pool of water, twisted, stretched, squashed, and is then dropped back into the pool. The control meshes for this sequence were sparse, and the per-frame control meshes were generated using our volume-preserving morpher.*

meshes for this sequence were already given per-frame, so our morpher was not used. These meshes are from the multi-view silhouette approach to mesh animation by Vlasic et al. [VBMP08]. Note that these input meshes do not conserve volume through the course of the animation, but our divergence correction managed to deal with this without introducing noticeable artifacts. This sequence, like that of the bunny, shows that the fluid closely tracks the control mesh, yet fluid is still allowed to escape in natural drips and sheets. This sequence also demonstrates the benefit of attracting the fluid surface to the control mesh. Even though this simulation was run at a resolution of $100^3$, the attraction of the fluid surface to the control mesh preserves many of the fine details of the control mesh.

In the accompanying video, we demonstrate surface waves in several variations of the head animation sequence shown in Figure 5. Like the dancer sequence, the control mesh for this sequence was provided on a per-frame basis, and in this case the meshes were produced by the single-view reconstruction method of Li et al. [LAGP09]. Our first sequence of the head shows an expression change from smile to frown, in zero gravity. The motion of the jaw produces ripples on the cheeks and across other portions of the face, and these ripples are purely due to our solution of the wave equation on the fluid surface mesh. This demonstrates that our control waves can track targets without resorting to any grid based force. Other sequences include gravity and use an animation of the head shaking back and forth. We demonstrate the effects of relaxed control (using a blend factor of 0.25) as well as those of mesh-based attraction and surface waves in Figure 5. A variety of behaviours ranging from tight tracking to drippy shapes with dynamic surface waves can be achieved by using a combination of these control parameters. Finally, we demonstrate how our control technique responds to external forces when a fast moving bunny collides with the animated head. In this example, we did not enable waves and used a blend factor of 0.2 for the relaxed control.

## 8. Limitations

There are a few limitations to our mesh-based approach to fluid control. If our volume preserving morpher is to be used, the input meshes must have the same vertex connectivity. We note that there are several published approaches for taking two existing meshes and creating vertex correspondences between them [LDSS99, KS04, SAPH04]. If our morpher is not required, then each control mesh can have a distinct connectivity from the others. In this case, we would also require per-vertex velocities to be provided.

A second limitation of our approach is that our control mesh sequences cannot move too fast. High velocities do not automatically destabilize the fluid simulation, but they can cause abrupt changes in wave heights as well as topological changes that may manifest themselves as surface artifacts or aliasing. The solution is to take smaller timesteps and linearly interpolate between the keyframes to obtain additional inbetween frames. This comes at the cost of introducing artifacts due to the interpolation.

## 9. Conclusion and Future Work

We have presented a system for controlling the motion of fluids, with a particular emphasis on creating and tracking specific shapes. The unifying theme of our approach is the control mesh that the animator provides in order to guide the fluid motion. If only a sparse set of such control meshes are given, then our system uses a volume preserving morph to provide a dense set of meshes. For a given time step, the control mesh is rasterized onto a grid, and this rasterization provides velocities that act as boundary conditions during the pressure projection step of fluid simulation. The fluid can strictly follow the control mesh, or the control can be more relaxed if the animator chooses. Finally, surface waves can be generated by attracting the fluid surface to the control mesh and also by solving the wave equation on the fluid surface.

There are several avenues for future work. One possibility is to use our control techniques for fluid simulation methods other than Eulerian grids, such as those methods that represent the fluid as tetrahedral elements. Also, since we have an explicit mesh for tracking the fluid surface, we could use this mesh to carry along foam and other materials on the fluid surface. Finally, our examples demonstrate the creation of moving 3D shapes in fluids, which can be used for film effects. We think that our method can also be used to create more subtle fluid control effects such as changing the shape of a breaking wave, but we have not yet tried this.

## 10. Acknowledgments

This work was partially funded by NSF grants CCF-0811485 and IIS-1130934. We would like to thank Scanline VFX for additional funding. We would like to thank Jie Tan as well as our anonymous reviewers for their useful suggestions and feedback.

## References

[AB97] AUBERT F., BECHMANN D.: Volume-preserving space deformation. *Computers and Graphics 21*, 5 (1997), 625 – 639. 3

[Ale02] ALEXA M.: Recent advances in mesh morphing. *Computer Graphics Forum 21*, 2 (2002), 173–196. 3

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A. K. Peters, 2008. 2, 6

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. *ACM Trans. Graph. 23* (August 2004), 441–448. 2

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graphical models and image processing 58*, 5 (1996), 471–483. 2

[FM97] FOSTER N., METAXAS D.: Controlling fluid animation. In *Proceedings of the 1997 Conference on Computer Graphics International* (1997), CGI '97, pp. 178–188. 2

[GHN01] GOULD N. I. M., HRIBAR M. E., NOCEDAL J.: On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM J. Sci. Comput. 23*, 4 (Apr. 2001), 1376–1395. 4

[GMSW87] GILL P. E., MURRAY W., SAUNDERS M. A., WRIGHT M. H.: *A Schur-Complement Method for Sparse Quadratic Programming*. Tech. Rep. SOL 87-12, Stanford University, Systems Optimization Laboratory, December 1987. 4

[GVL96] GOLUB G. H., VAN LOAN C. F.: *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*, 3rd ed. The Johns Hopkins University Press, Oct. 1996. 4

[HK04] HONG J.-M., KIM C.-H.: Controlling fluid animation with geometric potential. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 147–157. 2

[HMK11] HUANG R., MELEK Z., KEYSER J.: Preview-based sampling for controlling gaseous simulations. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM, pp. 177–186. 2

[HML99] HIROTA G., MAHESHWARI R., LIN M. C.: Fast volume-preserving free form deformation using multi-level optimization. In *Proc. Symposium on Solid Modeling and Applications* (1999), ACM Press, pp. 234–245. 3

[KLL*07] KIM B., LIU Y., LLAMAS I., JIAO X., ROSSIGNAC J.: Simulation of bubbles in foam with the volume control method. In *ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 6

[KS04] KRAEVOY V., SHEFFER A.: Cross-parameterization and compatible remeshing of 3d models. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 861–869. 9

[LAGP09] LI H., ADAMS B., GUIBAS L., PAULY M.: Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 175. 9

[LDSS99] LEE A., DOBKIN D., SWELDENS W., SCHRÖDER P.: Multiresolution mesh morphing. In *Proceedings of SIGGRAPH* (1999), vol. 99, pp. 343–350. 9

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. *ACM Trans. Graph. 23* (August 2004), 449–456. 2

[NB11] NIELSEN M. B., BRIDSON R.: Guide shapes for high resolution naturalistic liquid simulation. In *ACM SIGGRAPH 2011 papers* (2011), SIGGRAPH '11, pp. 83:1–83:8. 2

[NCZ*09] NIELSEN M. B., CHRISTENSEN B. B., ZAFAR N. B., ROBLE D., MUSETH K.: Guiding of smoke animations through variational coupling of simulations at different resolutions. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 217–226. 2

[REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMNER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), SCA '04, pp. 193–202. 2

[SAPH04] SCHREINER J., ASIRVATHAM A., PRAUN E., HOPPE H.: Inter-surface mapping. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 870–877. 9

[Sta99] STAM J.: Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), pp. 121–128. 2

[SY05a] SHI L., YU Y.: Controllable smoke animation with guiding objects. *ACM Transactions on Graphics 24* (January 2005), 140–164. 2

[SY05b] SHI L., YU Y.: Taming liquids for rapidly changing targets. In *ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), SCA '05, ACM, pp. 229–236. 2

[TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. In *ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), pp. 7–12. 2

[TWGT10] THÜREY N., WOJTAN C., GROSS M., TURK G.: A multiscale approach to mesh-based surface tension flows. *ACM Transactions on Graphics 29* (July 2010), 48:1–48:10. 2, 6, 8

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 97. 9

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. *ACM Trans. Graph 25* (2006), 1118–1125. 3

[WTGT10] WOJTAN C., THÜREY N., GROSS M., TURK G.: Physics-inspired topology changes for thin fluid features. *ACM Transactions on Graphics 29* (July 2010), 50:1–50:8. 2, 3, 6, 7