Interactive Localized Liquid Motion Editing

Zherong Pan[1] Jin Huang[1] Yiying Tong[2] Changxi Zheng[3] Hujun Bao[1]*
[1]State Key Lab of CAD&CG, Zhejiang University [2]Michigan State University [3]Columbia University

Abstract

Animation techniques for controlling liquid simulation are challenging: they commonly require carefully setting initial and boundary conditions or performing a costly numerical optimization scheme against user-provided keyframes or animation sequences. Either way, the whole process is laborious and computationally expensive.

We introduce a novel method to provide intuitive and interactive control of liquid simulation. Our method enables a user to locally edit selected keyframes and automatically propagates the editing in a nearby temporal region using geometric deformation. We formulate our local editing techniques as a small-scale nonlinear optimization problem which can be solved interactively. With this uniformed formulation, we propose three editing metaphors, including (i) sketching local fluid features using a few user strokes, (ii) dragging a local fluid region, and (iii) controlling a local shape with a small mesh patch. Finally, we use the edited liquid animation to guide an offline high-resolution simulation to recover more surface details. We demonstrate the intuitiveness and efficacy of our method in various practical scenarios.

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

Keywords: fluid simulation, sketch, deformation

1 Introduction

While fluid simulation has long been used to create realistic fluid animations, generating fluid animations with desired effects remains difficult and time-consuming. Liquid motion simulated with set initial and boundary conditions are intrinsically chaotic, making it hard to produce desired later frames by tuning parameters [Foster and Metaxas 1997a]. In computer graphics applications such as film production, controlling a liquid animation often requires multiple attempts even for experienced animators to obtain the desired results.

As a viable alternative to explicitly tuning parameters and boundary conditions, some previous methods [Treuille et al. 2003; McNamara et al. 2004; Wojtan et al. 2006] proposed to specify *a priori* a set of global fluid shape keyframes and rely on an offline numerical optimization scheme to find the desired fluid animation. However, unlike solid objects, authoring even a single fluid keyframe is laborious, and the manually designed frames tend to be non-volume-preserving and overly smooth, suppressing rich visual details in natural fluid motion. Moreover, the nonlinear optimization process over the dynamics with provided keyframes is computationally expensive. It is very hard, if not impossible, to provide a user fast feedback of resulting effects of the supplied keyframes.

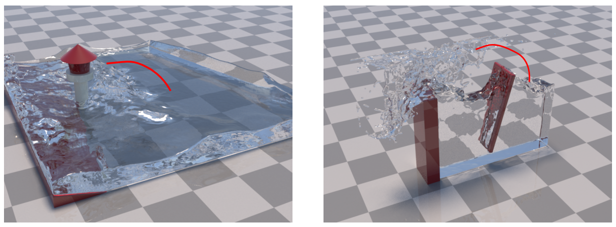
Figure 1: *The user can guide the fluid simulation by sketching (red curves) for the desired motion with fast feedback, crucial to animation prototyping. From the left to the right: a lighthouse at shore in tidal waves; a wave raised to engulf the lighthouse; a washing machine in action; water made to splash into anther chamber at a chosen instant.*

Instead of globally keyframing the entire fluid shape, we propose a method to edit fluid animation in a *local* spatial and temporal region. Limiting user editing in a local region enables an intuitive control process. Indeed, the general philosophy of local control has proven useful in many areas of computer graphics, such as spline curves [Barsky and Beatty 1983], shape editing [Botsch and Kobbelt 2005], and animation design [Cohen 1992]. For fluid control, this means that the user can progressively edit fluid animation toward the desired effects as the fluid simulation advances. Meanwhile, we can limit the size of the formulated optimal control problem, making an interactive feedback possible and hence accelerating the animation design cycles.

In our pipeline, the user starts from either a low-resolution or a downsampled high-resolution fluid simulation. While the simulation advances, one can select any resulting frame for editing. We provide three local editing metaphors: (i) the user can sketch a few strokes to guide the fluid local shapes or silhouettes projected on a view-dependent plane; (ii) for more detailed control, one can select a small fluid region and drag it locally around; and (iii) one can provide a small mesh patch to control local fluid features. All three metaphors boil down to a uniform optimization problem of *geometric* deformation, which deforms the fluid shape at the selected frame in an interactive and volume-preserving fashion. Once the selected frame is edited, its changes are then propagated over its previous frames to ensure temporally coherent results. Through guided simulation over a short local sequence, the newly edited frame also serves as updated initial conditions at the original resolution to affect subsequent fluid simulation. As the simulation progresses, the user incrementally edits the whole animation sequence. Alternatively, the entire sequence can be edited with low-resolution simulation, followed by an offline guided high-resolution simulation (e.g., [Nielsen and Bridson 2011; Yuan et al. 2011]) recovering fine details while still preserving desired motion effects.

Contributions Our approach features the following novel components:

- **Editing Metaphors:** we propose a versatile partial fluid keyframe editing interface allowing intuitive liquid surface modification through sketching, localized fluid dragging, and mesh-constrained local feature control.

- **Optimization Formulation:** we formulate all three editing metaphors using a highly efficient and unified optimization procedure (see §4).
- **Fast Preview:** we provide an interactive preview of edited animation by seamlessly blending the keyframe editing into a local temporal subsequence (see §5.1).
- **Guided Simulation:** to generate a final local subsequence, we propose a guided simulation method that produces physics and meanwhile respects edited effects (see §5.2).

2 Related Work

There have been numerous works on fluid simulation and animation control. Those most closely related to our work span cross three main areas: fluid control, shape deformation and sketch-based modeling.

Animating fluid flows goes back to the early procedural method [Kajiya and Von Herzen 1984] and the simple linear water flow model [Kass and Miller 1990]. Navier-Stokes fluid model has been popular in computer graphics since the early work [Foster and Metaxas 1996; Foster and Metaxas 1997b; Stam 1999; Enright et al. 2002]. In this paper, our fluid simulator is based on FLIP method [Zhu and Bridson 2005], since its concept of fluid particles eases the formulation of our control problems.

In parallel to the development of fluid simulation techniques, researchers have strived to address the difficulty of editing simulations, starting by the initial work for controlling pool balls [Barzel et al. 1996] and later general rigid bodies [Gleicher 1997; Popović et al. 2000; Chenney and Forsyth 2000]. Recent work [Huang et al. 2011; Barbič et al. 2012; Li et al. 2013] made it possible to interactively edit the motion of deformable bodies as well. In addition, Kircher and Garland [2006] also proposed a method to enhance the details of an existing deformable surface. However, techniques for controlling fluid animations are far less matured. Initial work by Foster and Metaxas [1997a] proposed a high-level user control over fluid parameters. Later, Treuille et al. [2003; 2004] applied optimal control over an Eulerian fluid animation with user-supplied fluid shape keyframes. To avoid expensive optimization over the entire fluid sequence, empirical forces [Fattal and Lischinski 2004] and filament basis [Angelidis et al. 2006] were applied to smoke simulation, although it is not clear how they can be extended for liquid simulation. In all these methods, a commonly used control interface requires user-supplied keyframes to define entire fluid shapes at certain times. However, authoring fluid-like keyframes *a priori* is challenging: multiple tuning cycles coupled with offline optimization are often needed, resulting in a time-consuming and laborious process. In contrast, we take a natural combination of physical simulation and manual crafting: the user can freely edit selected keyframes automatically created by an underlying fluid simulation; partial keyframe editing is realized by different control metaphors; and our geometric deformation enables interactive feedback in a local temporal region.

Instead of providing keyframes, a dense sequence of control meshes can be used to guide the fluid simulation through the methods such as [Shi and Yu 2005; Raveendran et al. 2012]. However, they provide no feedback to users until the re-simulation is finished. Recent methods [Nielsen and Bridson 2011; Yuan et al. 2011] have also aimed to loosely guide a high-resolution simulation with a provided low-resolution fluid animation. Complementary to all these methods which assume a desired mesh sequence or low-resolution fluid animation is prepared as input, we address the problem of creating a desired low-resolution fluid animation interactively. Optionally, the above methods can be used to guide a high-resolution liquid simulation to recover more surface details.

Lastly, procedural methods [Kim et al. 2008; Schechter and Bridson 2008; Narain et al. 2008] have been explored to generate stochastic motions such as turbulence flows. Bridson et al. [2007] proposed curl noise as a tool to model divergence-free fluid without explicitly simulating the physics. Our geometric fluid deformer is built upon this idea to provide fast feedback on a local fluid subsequence.

In the area of deformable solids, surface deformation techniques have proven successful to generate natural-looking surfaces satisfying user constraints. Most of the methods [Botsch and Sorkine 2008] focused on achieving elastic effects for solids, in which local deformation is limited as small as possible. However, this assumption breaks down for fluid deformation due to the lack of shearing resistance. Von Funck et al. [2006] proposed a fluid-like deformation technique that interactively deforms a model through a set of divergence free vector fields in analytical forms. A similar idea is applied in [Angelidis and Singh 2007] for skinning animation. In contrast to our method, this work has completely different goals: it aims to achieve mesh surface deformation that is volume-preserving and free of self-intersections, and by no means optimize deformation vector field to match use-specified targets.

Finally, in geometric modeling, a recent trend towards building intuitive user interfaces leads to the emergence of sketch-based methods, which automatically create 3D models based on 2D freehand drawings. We refer the reader to the paper [Olsen et al. 2009] for a comprehensive survey. Sketch-based methods can also be used to construct entire shapes or augment details [Igarashi et al. 1999]. The lack of precise description in sketches is often considered a weakness, but, as demonstrated by Lee et al. [2011], it has the ability to diversify the resulting models from sketched shape outlines or silhouettes. Related to our work, the idea of using sketches to specify user editing has been explored for both shape deformation [Nealen et al. 2005; Kho and Garland 2005] and image editing [Eitz et al. 2007]. However, it is quite difficult to directly extend these sketch-based deformation methods to liquid animations. Fluid shapes are generally more complex, and the control must preserve volume and maintain temporal coherence, necessitating the development of a new sketch-based fluid control method.

3 Overview

An overview of our algorithm pipeline is depicted in Figure 2. As we focus on fast prototyping of fluid animation, we start from a relatively low-resolution (or downsampled high-resolution) fluid simulation. While our method does not critically depend on any particular fluid simulation method, we employ the FLIP/PIC method [Zhu and Bridson 2005], a hybrid Eulerian-Lagrangian solver, since its Eulerian and Lagrangian representations of fluids provide the flexibility for processing user sketches. User editing begins when a user pauses the simulation and selects a simulated frame. Our animation design pipeline consists of two major stages, *keyframe design* and *sequence generation*. In the keyframe design stage, a user interactively modifies the shape of selected keyframe using sketches, dragging or providing local mesh patches, all of which are formulated into a uniform optimization problem. In sequence generation stage, the editing on the keyframe is propagated backward to a local subsequence for a realtime preview of resulting animation. This propagation respects the dynamics in the original sequence, and can be optionally controlled by a temporal control curve. Finally, the local subsequence is merged into the original sequence and the simulation resumes with the deformed keyframe as the initial condition

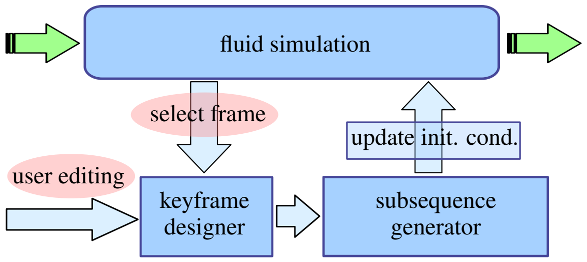

Figure 2: Overview: *The green arrows indicate the start and end of our pipeline. It begins with a low-resolution fluid simulation. The user at any point can select a simulated frame to edit using three different control metaphors (see Section 4). The edited frame becomes a keyframe, followed by corresponding adjustments over a local subsequence of frames to ensure temporal coherence (see Section 5. The newly created keyframe then serves as updated initial conditions for subsequent simulation. Finally, a high-resolution simulation is guided by the edited low-resolution simulation to recover more surface details (see Section 5). Red ellipses indicate user interaction to the pipeline.*

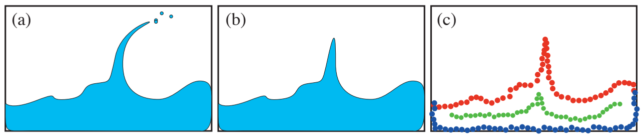

Figure 3: *(a) The liquid shape at the selected frame contains tiny splashes. (b) These splashes are discarded by extracting the main body of liquid. (c) Three fluid particle subsets are extracted (red for free surface particles, blue for solid boundary particles and green for medial axis particles.*

for the subsequent simulation.

Notation In the following presentation, we use bold upper-case letters to represent vectors and functions, a calligraphic font (e.g., $\mathcal{S}$) to denote a set, and a sans-serif font (e.g., D) to indicate a deformation operator. At each frame i, its fluid shape is represented by a set, $\mathcal{P}^i$, of fluid particles. In our FLIP/PIC fluid simulation, $\mathcal{P}^i$ is just the fluid particles used in the current simulation, and the corresponding user-modified fluid shape is represented by $\bar{\mathcal{P}}^i$.

4 Keyframe Designer

Our underlying fluid simulation generates a liquid animation represented by a sequence of fluid particle sets, $\mathcal{P}^i$. The goal of our keyframe designer is to allow a user to interactively edit liquid shapes at selected keyframes. For this purpose, we provide three editing metaphors: (i) sketching a few strokes to specify desired fluid shape, (ii) dragging a small region to locally deform the fluid, and (iii) providing a small mesh patch to constrain the local shape. In this section, we mainly present how we formulate sketch-based control as a small optimization problem that can be solved interactively (see Algorithm 1). The other two editing metaphors are formulated in similar ways, and we only highlight the necessary differences in their formulations.

```
Algorithm 1: Keyframe Design by Stroke Matching
  Sample each stroke to construct a set of control particles $\mathcal{L}$.
  Optimize the velocity field $V$ (see §4.2).
  Discretize $V$ on the grid.
  Advect $\mathcal{P}^K$ along $V$ to get $\bar{\mathcal{P}}^K$.
```

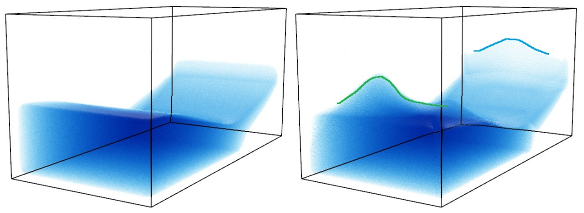

Figure 4: *By using two types of shape particles $\mathcal{B}$ and $\mathcal{S}$, we allow users to modify boundary location and free surface respectively. For example, to get the desired shape on the right, both the solid boundary (blue stroke) and free surface (green stroke) should be modified.*

Fluid Shape Preprocessing Provided a user-selected frame K, first we discard small features (e.g., thin splashes) of the liquid shape, since those detailed features often introduce complex occlusions for user sketching while contributing little for the overall fluid shape control. Specifically, We simply apply a morphological opening operation which erodes the level-set field (constructed from the particle set $\mathcal{P}^K$) by 3 grid spacing and then dilates by 3 grid spacing. We note that a similar approach has been used in [Nielsen and Bridson 2011]. Next, from the resulting liquid shape, we extract three representative sets of fluid particles: (i) the solid boundary (solid-liquid interface), $\mathcal{B}^K$, (ii) the free surface (liquid-air interface), $\mathcal{S}^K$, and (iii) the medial axis (for 2D) or medial surface (for 3D), $\mathcal{M}^K$. All three sets are subsets of $\mathcal{P}^K$, but distinguishing the boundary particles in different types allows a user to clarify the semantic ambiguity of sketches (see Figure 4). We extract $\mathcal{B}^K$ and $\mathcal{S}^K$ based on the level-set values; we estimate $\mathcal{M}^K$ following the approach [Adams et al. 2007]. We refer the reader to Figure 3 for an illustration of the preprocessing step. From now on, we drop the superscripts of the set notations for simplicity, since the presentation is based on the current selected frame K.

From User Sketch to Control Particles Corresponding to the three subsets of representative particles (namely, $\mathcal{B}$, $\mathcal{S}$ and $\mathcal{M}$), we provide a user three types of strokes to specify the desired shape of each kind of particles. After the user sketches strokes on the 2D screen, we transform their coordinates into the simulation space, and then generate corresponding *control particles* $\mathcal{L}$. For 2D simulation, the above procedure amounts to a simple scaling and sampling from the curves. In 3D simulation, however, we need to (i) determine the depth of the stroke with respect to the current viewpoint, (ii) form a patch by extruding from it along viewing direction, and (iii) then sample the control particles from the patch. We hypothesize that the desired depth usually aligns with the closest salient features of the fluid shape. Without loss of generality, suppose that a user stroke is to specify the boundary shape. We cast a set of rays $\mathcal{R}$ from the camera (located at $\boldsymbol{e}$) toward each control particle in $\mathcal{L}$ and find a uniform depth d_s by minimizing the total distance to $\mathcal{B}$, i.e., $d_s = \arg\min_d \sum_{\boldsymbol{r}\in\mathcal{R}} \min_{\boldsymbol{p}\in\mathcal{B}} \mathsf{dist}(\boldsymbol{e} + \boldsymbol{d_r} d, \boldsymbol{p})$, where $\boldsymbol{d_r}$ is the direction of the ray. Note that $\min_{\boldsymbol{p}\in\mathcal{B}} \mathsf{dist}(\boldsymbol{e} + \boldsymbol{d_r} d, \boldsymbol{p})$ for any given d is found by directly accessing the distance field of $\mathcal{B}$. It is possible to use one distance per stroke, but we follow the suggestion by Zimmermann et al. [2008], which handles a similar problem of finding salient feature on a surface mesh. Our simple solution results in more robust and satisfactory depth estimation for implicit represented and commonly ambiguous fluid shapes. Finally, we lift the 2D stroke to a 3D curve followed by extruding it along the viewing directions both forward and backward to form a small patch. Now our control particles $\mathcal{L}$ consist of sampled points on the extruded patch. The width W of the patch is set as a user-specified parameter to control the influence range of a stroke (see Figure 5 and the video for an illustration).

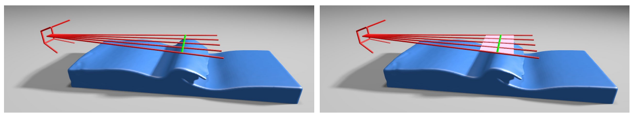
Figure 5: Determining depth of a 3D stroke: *Left: Closest liquid feature is found through exhaustive line searching and 2D sketch is lifted to 3D (green) curve. Right: The 3D curve is extruded in both directions to form a (purple) patch.*

4.1 Objective Function of Control Particles

After laying out the preprocessing details, we are now ready to formulate user sketching control as an optimization problem. Without loss of generality, we only detail our algorithm for editing fluid boundary shape $\mathcal{B}$. The free surface and medial axis editing are formulated in exactly the same way. In this subsection, we introduce our objective function based on boundary particles $\mathcal{B}$ and the control particles $\mathcal{L}$ defined by user sketches. Our goal is to construct a deformation operator D that deforms the liquid shape such that the deformed boundary particles $\bar{\mathcal{B}}$ matches as close as possible to the shape defined by $\mathcal{L}$.

Perhaps one simple objective function can be defined as

$$\Theta_0(\mathcal{L}, \mathcal{B}, \mathsf{D}) = \sum_{\mathbf{B}_i \in \mathcal{B}} \phi^2_{\mathcal{L}}(\mathsf{D}(\mathbf{B}_i)), \tag{1}$$

where $\phi_{\mathcal{L}}$ is the distance field computed from $\mathcal{L}$. Unfortunately, this formulation is computationally expensive since it has to iterate over a potentially large set of boundary particles $\mathcal{B}$ and apply the deformation operator D on each particle. Moreover, this function penalizes the distance value of all boundary particles, but in most cases a user just wants to enforce partial constraints specified by sketching. To resolve these shortcomings, we propose a different objective function

$$\Theta(\mathcal{L}, \mathcal{B}, \mathsf{D}) = \sum_{\mathbf{L}_i \in \mathcal{L}} \phi^2_{\mathcal{B}}(\mathsf{D}^{-1}(\mathbf{L}_i)), \tag{2}$$

where D^{-1}, the inverse operator of D, needs to be evaluated only on $\mathcal{L}$ which consists of much fewer number of particles than $\mathcal{B}$ does. In addition, it penalizes only a subset of $\mathcal{B}$, avoiding the fictitious global deformation guidance from $\phi_{\mathcal{L}}$.
Remark. When a mesh patch is used to provide partial control over geometric details, the same formulation can be used with $\mathcal{L}$ created from the samples on the mesh surface. For dragging a local region from the start location $\boldsymbol{s}$ to a target location $\boldsymbol{t}$, we use a similar yet simpler objective function

$$\Theta(\boldsymbol{t}, \boldsymbol{s}, \mathsf{D}) = \|\mathsf{D}^{-1}(\boldsymbol{t}) - \boldsymbol{s}\|^2. \tag{3}$$

The depth of the dragging line is determined in the same way as we determine a stroke depth presented earlier.

4.2 Optimizing the Deformation Operator

With the defined objective function, we now find an optimal deformation operator D to minimize it. The challenge is to find D in realtime while ensuring natural deformation of the liquid shape. Although many methods have been proposed for elastic deformation [Botsch and Sorkine 2008], liquid has fundamentally different characteristics due to the lack of shearing resistance, the property of volume preservation, and its complex boundary conditions, rendering prior solid shape editing methods inapplicable for our purpose.

Reduced Model of Deformation Operator Our key idea of constructing D is to deform the liquid shape by advecting fluid particles $\mathcal{P}$ under a steady velocity field $\boldsymbol{V}$ with N_A (forward Euler) steps. In other words, D is defined as $\mathsf{D}(\mathbf{P}) = \mathsf{Adv}^{N_A}(\mathbf{P}, \mathbf{V})$ for $\mathbf{P} \in \mathcal{P}$, where $\mathsf{Adv}(\cdot, \mathbf{V})$ denotes advection by $\mathbf{V}$ for a single forward Euler step. Then the corresponding inverse operator D^{-1} in the objective function (2) is

$$\mathsf{D}^{-1}(\mathbf{P}) = \mathsf{Adv}^{N_A}(\mathbf{P}, -\boldsymbol{V}).$$

Now, we need to optimize over $\boldsymbol{V}$ to minimize Θ. However, direct representation of $\boldsymbol{V}$ has too many degrees of freedom to allow realtime optimization. A logical strategy is to use a reduced model of $\boldsymbol{V}$ to shrink the problem size. Previous fluid model reduction methods [Treuille et al. 2006] extracting a reduced bases from training examples require a large memory footprint due to the lack of analytical representations. Therefore, they are impractical to be used in our realtime fluid deformers. In addition, the property of global support in those basis functions loses the advantages of local control, leading to unpleasant artifacts in the region far from the constrained locations. Instead, we adapt the curl noise model proposed in [Bridson et al. 2007], which nicely meets our requirements for sketch-based local editing.

Concretely, since the curl of an arbitrary vector field $\mathbf{H}$ (i.e., $\boldsymbol{V} = \nabla \times \mathbf{H}$) is inherently solenoidal, we first define a reduced model of the potential field $\mathbf{H}$ and use it in our optimization. We use free-slip solid boundary condition for all our examples. Thus, the normal component of $\mathbf{H}$ is left unchanged, while the tangential component of $\mathbf{H}$ is set to zero on the boundary, which corresponds to the no transfer condition (i.e., vanishing normal component of $\boldsymbol{V}$). By smoothly reducing the tangential component of $\mathbf{H}$ approaching solid boundary, we have

$$\mathbf{H}_R = \begin{cases} \mathbf{H}\ \mathbf{R}(\phi/\epsilon) + (1 - \mathbf{R}(\phi/\epsilon))\mathbf{N}(\mathbf{H}\cdot\mathbf{N}) & \text{for 3D cases,} \\ \mathbf{H}\ \mathbf{R}(\phi/\epsilon) & \text{for 2D cases.} \end{cases}$$

Here ϕ is the level set function of solid boundary, and ϵ is the boundary ramping layer thickness. Similar to [Bridson et al. 2007], the ramp $\mathbf{R}(\cdot)$ is defined as

$$\mathbf{R}(\psi) \quad = \quad \begin{cases} 1 & \psi \geq 1, \\ \frac{15}{8}\psi - \frac{10}{8}\psi^3 + \frac{3}{8}\psi^5 & 1 > \psi > -1, \\ -1 & \psi \leq -1. \end{cases} \tag{4}$$

The specific form of $\mathbf{H}$ encodes the body forces for driving the keyframe shapes. In practice, we use two force bases: the curl force (vortex-like) and the wind force (wind in one direction):

$$\mathbf{H}_C(\mathbf{X}, \mathbf{C}, \mathbf{A}, f, f_0) \quad = \quad \mathbf{A}E, \tag{5}$$

$$\mathbf{H}_W(\mathbf{X}, \mathbf{C}, \mathbf{A}, f, f_0) \quad = \quad \mathbf{A} \times (\mathbf{X} - \mathbf{C})E, \tag{6}$$

where $E = \exp(-(f^2 + f_0)\|\mathbf{X} - \mathbf{C}\|)$, $\mathbf{C}$ is the center of force, f and f_0 are the fade parameters, and $\mathbf{A}$ is the axis or direction of force. The total potential field $\mathbf{H}$ is the summation of all the N force fields specified by their individual parameters

$$\mathbf{H} = \sum_{i}^{N} \mathbf{H}^i_{c_i}, \quad c_i \in \{C, W\}, \tag{7}$$

where c_i is the type of the force model for force field i. In summary, with this potential field model, the velocity field $\boldsymbol{V} = \nabla \times \mathbf{H}$ is expressed using the variables $\mathbf{C}$, $\mathbf{A}$, f and f_0 for each basis model. Our goal now is to solve for these variables to minimize Θ in the objective function (2).

Regularization of Objective Function To achieve stable results and avoid overfitting, we consider two additional regularization terms. The first term $\mathbf{E}_{reg}$ penalizes the displacement of control particles before and after advection:

$$\mathbf{E}_{reg} = \sum_{\mathbf{L}_i \in \mathcal{L}} \|\delta \mathbf{L}_i\|^2, \tag{8}$$

where $\delta\mathbf{L} = \mathsf{D}^{-1}(\mathbf{L}) - \mathbf{L}$. The second term $\mathbf{E}_{lap}$, similar to [Sorkine et al. 2004], is the Laplace regularization applied on $\mathcal{L}$. Namely,

$$\mathbf{E}_{lap} = \sum_{\mathbf{L}_i \in \mathcal{L}} \Big\| \sum_{\mathbf{L}_j \in \mathcal{N}(\mathbf{L}_i)} w_{ij}(\delta\mathbf{L}_j - \delta\mathbf{L}_i) \Big\|^2, \tag{9}$$

where $\mathcal{N}(\mathbf{L}_i)$ stands for the set of neighboring control particles of $\mathbf{L}_i$ (those within the support of its kernel), and $w_{ij} = \frac{1}{|\mathcal{N}(\mathbf{L}_i)|}$.

Remark. For mesh-based control particles, we replace $\mathcal{N}(\mathbf{L}_i)$ by the one-ring neighborhood of the mesh vertex $\mathbf{L}_i$, and set w_{ij} to the cotan weights used in the usual discrete Laplacian [Meyer et al. 2002].

Our final objective function is

$$\mathbf{E}_{final} = \Theta + w_r \mathbf{E}_{reg} + w_l \mathbf{E}_{lap}, \tag{10}$$

where the two weights, w_r and w_l, are user-controllable parameters. w_r controls the amount of stablization; w_l controls the matching pattern of liquid body: higher weight produces smoother and looser matching results with larger influence range, while lower weight leads to less smooth but tighter and more accurate matching. Notice that our basis is free to move in space and to scale in size. Therefore, in practice, very few force fields are needed to match the user sketches reasonably well (typically $N = 10$ in (7)). Solving the force field parameters to minimize $\mathbf{E}_{final}$ determines the velocity field $\boldsymbol{V}$ for liquid shape deformation. We then advect all the liquid particles $\mathcal{P}$ along $\boldsymbol{V}$ using multiple forward advection steps. In following description, we refer this advection process as *post-warping*.

Interactive Numerical Solver Although the formulated optimization problem (2) is nonlinear, its complexity scales according to the sketch size rather than the complexity of liquid body, and the size of our reduced model is quite small (typically $N = 10$). In practice, we can easily solve it using a dense Levenberg-Marquardt (LM) solver [Press et al. 2007]. Also notice that the processed particles are limited in a small set $\mathcal{L}$; all the summation terms in (10) and their derivatives required in LM have analytical expressions, and hence can be easily evaluated in parallel. We implement these calculations as well as post-warping for fluid deformation all on GPU, and the rest of the computation is carried out on CPU. Typically, the number of advection steps in post-warping can be much larger than the number of backward advection step (namely, N_A) used in the optimization. On the other hand, the cost of the optimization increases with N_A rapidly, so we opt to take a small N_A with a large step size. In our experiments, the result is insensitive to this parameter as long as it is no less than 10, and $N_A = 10$ is used in all our results. In Figure 6, we show the results using 7, 10 and 30 steps for N_A (see §6 for more implementation details). Putting all the pieces together eventually allows interactive performance of the optimization solver, and provides immediate feedbacks after user sketching.

5 Sequence Generator

In this section, we integrate the user-edited frame into the fluid sequence to form a temporally coherent animation.

5.1 Update of Local Subsequence

After the user edits a frame K, a local subsequence of frames prior to frame K, $\mathbb{S}^K = \{\mathcal{P}^{K-i} | 0 < i < w\}$, is selected, where w

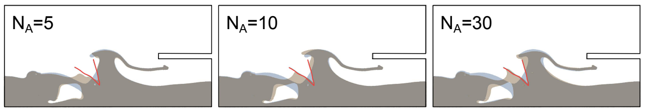

Figure 6: *Matching result of a V-Shaped free surface sketch (blue for the input shape and grey for the deformed shape). Comparison of different numbers of advection steps: $N_A = 5$, $N_A = 10$ and $N_A = 30$ respectively. Higher streamline resolution leads to better match against the user sketch but longer computation time. In practice, we found $N_A \geq 10$ produces plausible results.*

denotes the local window size ($w = 50$ by default). Next, our algorithm automatically adjusts these frames to ensure temporally coherent resulting sequence and provides the user with a preview in realtime. Recall that our deformation operator D at frame K is defined by the potential field $\mathbf{H}$ in Equation (7). Our plan is to deform each frame in $\mathbb{S}^K$, such that the amount of deformation smoothly increases to the full deformation D at frame K (see Algorithm 2 for an outline of this step).

Algorithm 2: Generate Deformation for Frame $K - i$

```
Advect force centers of $H$ backward along fluid velocity $\boldsymbol{v}$
Calculate signed distance $\phi$ in Equation (4) for $H_R$
Compute the deformation velocity field with ramping $F$
Advect $\mathcal{P}^{K-i}$ by the deformation velocity field
```

One simple way is to apply a ramp function on $\mathbf{H}$ to smoothly increase the magnitude of the force field. However, this idea completely ignores the underlying fluid dynamics and hence produces unnatural results. Instead, to respect the fluid dynamics, we propose to advect backward in time the force centers of field $\mathbf{H}$ by the input fluid velocity field $\boldsymbol{v}^K$ at each frame $i \in \mathbb{S}^K$ before ramping $\mathbf{H}$. Intuitively, this amounts to applying an increasing amount of deformation on approximately the same set of fluid particles as we do at frame K. In practice, the force centers resulting from the optimizer might locate outside of the fluid domain where the fluid velocity field $\boldsymbol{v}^i$ is undefined. For those cases, we simply extrapolate $\boldsymbol{v}^i$ to the entire simulation domain. Lastly, we use a ramp function $F : [0, 1] \to \mathbb{R}$ satisfying $F(0) = 0.0$ and $F(1) = 1.0$. At each frame $K - i, 0 < i < w$, we deform the particle set $\mathcal{P}^{K-i}$ into $\bar{\mathcal{P}}^{K-i}$ by post-warping with an amount of fictitious time attenuated by $F(1 - i/w)$. As shown in Figure 10, the user can intuitively control the ramp curve through a cubic curve editor.

5.2 Generation of Final Animation

Our fluid editing algorithm so far is highly efficient for realtime interactions. The interactive performance comes with our reduced geometric deformation operator D and fast local subsequence interpolation in section 5.1, but it pays a price of ignoring the underlying physics. Therefore, we retain fine details over the edited subsequence by running a guided local subsequence simulation, similar to the method in [Shi and Yu 2005]. Specifically, we run a short offline simulation for the frames indexed by $i, K - w < i \leq K$. To guide the simulation toward the user-edited liquid shape, we introduce a ghost force $\mathbf{f} = \mathbf{f}_{shape} + \mathbf{f}_{velocity}$. Here $\mathbf{f}_{shape}$ drives the source shape $\mathcal{P}^i$ toward the target shape $\bar{\mathcal{P}}^i$. The force is treated as a divergence free gradient field with the boundary condition

$$\mathbf{f}_{shape}(\mathbf{x}) = -C_{shape} \nabla \phi^2_{\bar{\mathcal{P}}^i}(\mathbf{x}), \quad \mathbf{x} \in \phi^{-1}_{\mathcal{P}^i}(0). \tag{11}$$

We solve a Laplace equation for its potential field with the above homogeneous Neumann boundary condition, and derive $\mathbf{f}_{shape}$ from the gradient of the resulting field. Here C_{shape} is a weighting parameter (500 in all examples). For $\mathbf{f}_{velocity}$, we follow Shi and Yu [2005] and use

$$\mathbf{f}_{velocity} = C_{velocity}(\boldsymbol{v}_{\bar{\mathcal{P}}^i} - \boldsymbol{v}_{\mathcal{P}^i}), \tag{12}$$

where $C_{velocity}$ is a weighting coefficient (with a default value 20), $\boldsymbol{v}_{\mathcal{P}^i}$ is the velocity field of the current frame, and $\boldsymbol{v}_{\mathcal{P}^i}$ is the target velocity.

A naive way to approximate $\boldsymbol{v}_{\bar{\mathcal{P}}^i}$ is applying finite difference between $\bar{\mathcal{P}}^{i-1}$ and $\bar{\mathcal{P}}^i$ for each particle. However, since our deformation procedure ignores the volumetric liquid motion in the subsequence, the internal fluid motion is usually incorrect, only the boundary shape is desired by the user. Therefore, we approximate the target velocity field $\boldsymbol{v}_{\bar{\mathcal{P}}^i}$ by solving another Laplace equation with the boundary condition

$$\boldsymbol{v}_{\bar{\mathcal{P}}^i}(\mathbf{x}) = \nabla\phi^2_{\bar{\mathcal{P}}^{i-1}}(\mathbf{x})/h, \quad \mathbf{x} \in \phi^{-1}_{\bar{\mathcal{P}}^i}(0), \tag{13}$$

where h is the time step. This formulation computes the velocity of boundary particles by their normal displacement between two consecutive frames, and interpolates the interior velocity. A final issue is that $\boldsymbol{v}_{\bar{\mathcal{P}}^i}$ so computed is irrotational, but a velocity field of fluid would not be irrotational in general. Thus the above formulation damps out the vorticity. To address this issue, we extract the potential part of $\boldsymbol{v}_{\{\mathbf{P}^i\}}$ through the operator $\mathbf{P}^*$, and use

$$\mathbf{f}_{velocity} = C_{velocity}(\mathbf{P}^*(\boldsymbol{v}_{\mathcal{P}^i})) - \boldsymbol{v}_{\bar{\mathcal{P}}^i}), \tag{14}$$

where $\mathbf{P}^*$ is applied by yet another Laplace equation with the boundary condition

$$\mathbf{P}^*(\boldsymbol{v}_{\mathcal{P}^i})(\mathbf{x}) = \boldsymbol{v}_{\mathcal{P}^i}(\mathbf{x}), \quad \mathbf{x} \in \phi^{-1}_{\mathcal{P}^i}(0). \tag{15}$$

As this uses the same boundary condition and computational domain as (11), both solves can be merged by summing up the right-hand side, introducing little overhead. In our experiments, the final solution faithfully respects the original sequence, while still effectively matching the target shape without overshooting. In summary, we only need two more Laplace solves for each frame in the local subsequence.

In the presence of splashes, applying ghost force on thin features introduces large artifacts. As we only need to control the overall shape, we extract the main fluid body as in the preprocessing stage. We compute the fluid signed distance field and only apply forces on particles belonging to the main fluid body.

Concatenating the sequence before $K - w$ with the above detail-enhanced subsequence, we resume the simulation on the rest of the frames to complete the sequence. As a recap, Algorithm 3 outlines our pipline for editing a user-selcected frame K.

Algorithm 3: Editing Iteration

```
Load a raw sequence.
Extract a subsequence $\mathbb{S}^K$ containing frame $K$.
Preprocess frame $K$.
Edit $\mathcal{P}^K$ using sketch [+mesh] to get $\bar{\mathcal{P}}^K$.
Deform subsequence $\mathbb{S}^K$ to get $\bar{\mathbb{S}}^K$ matching $\bar{\mathcal{P}}^K$.
Substitute $\bar{\mathbb{S}}^K$ into the raw sequence and continue simulation.
```

6 Implementation Details

In this section, we highlight a few implementation details critical for achieving interactive performance.

Memory Consumption One practical challenge lies in handling millions of particles in a typical 3D fluid sequence, as this greatly increases the cost of post-warping. We first perform an 8-to-1 downsampling when extracting selected frame and related subsequence. We notice that a simple downsample strategy that uses a coarse grid and chooses a random particle in each fluid grid leads to aliasing artifacts. In practice, we first use the grid cell centers as initial candidate particle positions, and then iteratively move the particles to the weighted average of nearby particles in the original particle set to remove the artifacts (we use 10 iterations in our tests). The data transfer between online editing and offline simulation stages is almost neglectable, since only the parameters of the potential field $\mathbf{H}$ is required to compute at a keyframe, and only a downsampled sequence is used for online editing. For better performance, we also deploy the offline simulation on a remote cluster, but keep the online editor residing on a commodity desktop PC for user interaction.

Partial Derivative Calculation The most costly part of the optimization is the partial derivative calculation. As mentioned earlier, the partial derivatives of our energy term bear a closed form except for the level set function, which is discretized in a standard Marker And Cell (MAC) grid and evaluated numerically. However, due to the complexity of our energy term, calculating the derivatives analytically takes an excessive amount of computational resources. Therefore, we use a mixed numerical-analytical formulation. Using the chain rule, we have

$$\frac{\partial \mathbf{E}}{\partial(\mathbf{C}, \mathbf{D}, f)} = \frac{\partial \mathbf{E}}{\partial \mathsf{D}^{-1}(\mathbf{P})} \frac{\partial \mathsf{D}^{-1}(\mathbf{P})}{\partial \mathbf{V}} \frac{\partial(\nabla \times \mathbf{H})}{\partial(\mathbf{C}, \mathbf{D}, f)}. \tag{16}$$

The first two terms in the right hand side are computed trivially in adjoint manner. The third term is costly to evaluate even by carefully applying a symbolic algebraic tool to simplify it. To reduce the cost, we compute the curl operator numerically using finite differencing with a spatial interval equal to the underlying grid spacing. This treatment keeps the same order of accuracy with the fluid solver. The other terms are calculated analytically.

Signed Distance Function from Solid Boundary In the force field models we used, the ramping function for the potential $\mathbf{H}$ depends on the signed distance ϕ from the solid boundary. However, it is well-known that the signed distance field involves discontinuities and shock lines (because the viscosity solution of Eikonal equation is not C^1 continuous). This can lead to discontinuous ramping function. To remedy this issue, we replace it with a smooth approximation of the signed distance function through R-functions introduced in [Pasko et al. 1995]. In particular, we first construct the solid boundary using Constructive Solid Geometry trees, through successive union and intersection operations on primitive shapes with no concave regions. Then we use the following R-functions to construct a signed field from the signed field of two solid shapes A and B,

$$\phi_{A\cup B} = \frac{\phi_A(x) + \phi_B(x) - \sqrt{\phi_A^2(x) + \phi_B^2(x) - 2\alpha\phi_A(x)\phi_B(x)}}{(1+\alpha)},$$

$$\phi_{A\cap B} = \frac{\phi_A(x) + \phi_B(x) + \sqrt{\phi_A^2(x) + \phi_B^2(x) - 2\alpha\phi_A(x)\phi_B(x)}}{(1+\alpha)},$$

where α is a smoothing factor, which leads to true distance function when $\alpha = 1$. In practice, we use $\alpha = 0.6$ to get smooth estimation near the solid boundary.

Realtime Visualization Our online editor requires a realtime visualization of fluid volume. To this end, we employ the screen-space curvature flow [van der Laan et al. 2009] for fluid volume visualization, avoiding the expensive surface mesh generation. Since

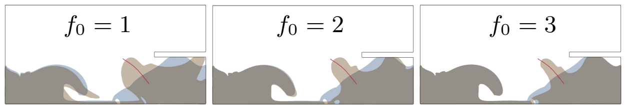

Figure 7: *A comparison of different locality. A higher value of f_0 leads to a smaller range of influence.*

this method was originally designed for SPH method but not the FLIP solver, we need to regulate the FLIP particles into an approximate Poisson distribution to ensure good visual quality. We achieve this using a modified FLIP solver as described in [Ando and Tsuruno 2011].

7 Results

In this section, we present first 2D examples to demonstrate the effects of our sketch types (Figure 11) and control parameters, then 3D results for intelligent sketch nodes generation from 2D screen sketch and finally a performance analysis.

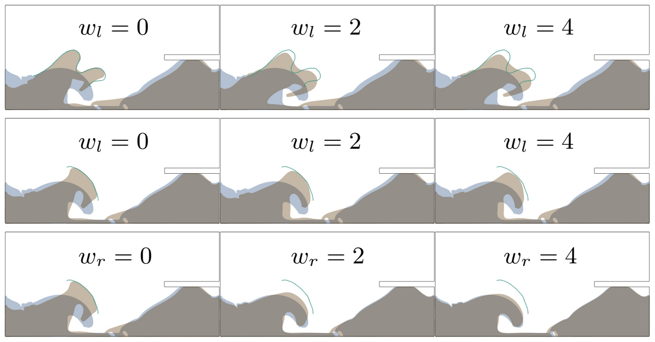

Figure 8: *A comparison of sketch matching effect under different Laplace or position weights. In the first row, the sketch shape is drastically different from that of the liquid body, in which case, larger Laplace panelty weight leads to smoother but coarser fitting. In the second row, the sketch shape matches the original local liquid feature well and the Laplace weight has neglectable effect. The third row shows the effect of position panelty, which regulates the resemblance to the original shape.*

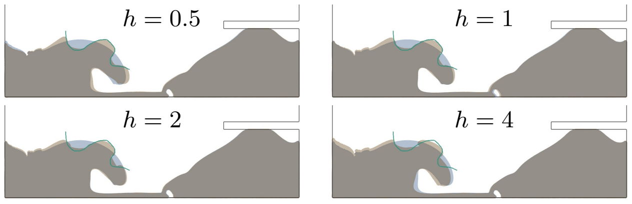

Figure 9: *A comparison of different number of force fields. We put one force field every h grid edge length. More force fields lead to more accurate matching but the difference is neglectable.*

One fading parameter (f) is automatically optimized whereas the other (f_0) is user controllable for desired locality. As shown in Figure 7, a larger value makes the sketch deform the shape more locally. A hand-drawn sketch itself is often inaccurate, and thus the desired smoothness and the extent of fitting should be balanced. Two regularization parameters can be adjusted to this end. Figure 8 shows the effects of the weighting on Laplace and position penalty terms, which try to maintain the smoothness of the shape control and the original shape of liquid body, respectively.

A large number of force fields provide more degrees of freedom in the optimization, and result in better matching accuracy, at an increased computational cost. With the desired smoothness of the deformation, a large number of force fields are often unnecessary, and may even lead to overfitting. In Figure 9, we show a side-by-side comparison of the deformation under different numbers of force fields. For a keyframe with several strokes, we sample the sketch at an interval of h, the grid spacing, and use one force field for each node.

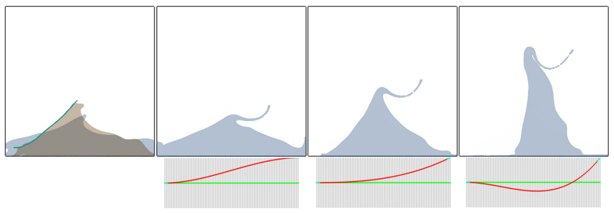

Figure 10: *We illustrate the effects of the tangent of the fading curve $F(x)$ at $x = 1$ by comparing the shape of liquid body 30 frames after the keyframe. As the tangent increases from left to right, more energy is injected into the system.*

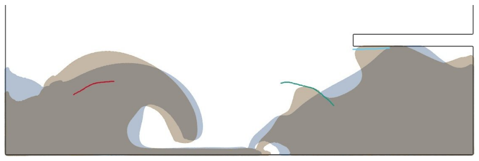

Figure 11: *The three types of 2D sketch strokes applied simultaneously: green for free surface sketch, red for medial axis sketch and blue for solid boundary sketch.*

One additional user control over the dynamics is the fading function F in local sequence generation. The slope of F at the keyframe is of most importance in design. It controls how much energy is introduced into the fluid system when matching the keyframe. In Figure 10, we compare the effects of different slopes.

In 3D cases, compared with directly placing 3D control particles to deform the shape, controlling the result by a 2D sketch is much more covenient and efficient (Figures 12 and 13). Key to intuitiveness is how to properly infer a set of 3D sketch nodes from it. In Figure 14, we show two typical cases of our sketch depth estimation mentioned in Section 4. The patch width is an additional use-specified parameter as is shown in Figure 15.

In the following, we demonstrate with several complex results how our method can be used to create storytelling animations intuitively and efficiently. Please refer to the accompanying video for the animation.

In the washer example (Figure 16), we wish to create the effect of water splashing from the main chamber into the neighboring chamber at a given time. Unlike previous optimal control based method, the user can draw a sketch and adjust the results (in a low resolution version) interactively to make sure the shape and amount of the splash is as desired. One may alternatively create a mesh to specify the global shape of the liquid surface at the keyframe, which would be not only costly to optimize, but also difficult to blend with the previous frames naturally. The convenience of sketch is further demonstrated in the wave invoking example (Figure 17), where a single sketch stroke suffices to create the effect of a Poseidon-like character raising a high wave to engulf the lighthouse with the desired violence. Finally, in the waterfall example (Figure 18), water is pulled into lower boxes sequentially by applying two consecutive editing passes with sketches.

Besides the sketch, more directed and detailed control can be pro-

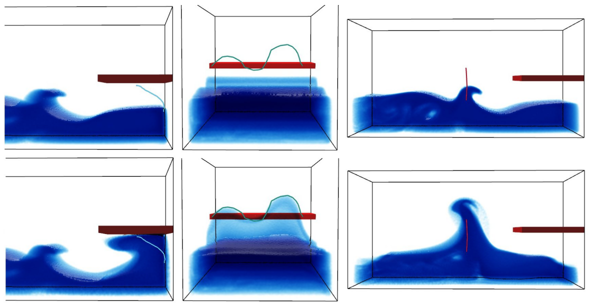

Figure 12: *The three types of strokes in 3D case. Top: the original shape. Bottom: the deformed shape matching the sketch stroke.*

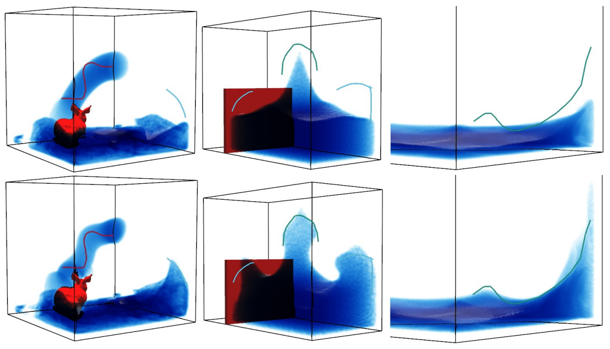

Figure 13: *3D sketch results. Top: original. Bottom: deformed.*

vided by placing mesh patches to shape the liquid surface in regions of interest, while still allowing other regions to be decided by the equations of motion. As shown in the dropping faces result (Figure 19), the droplets are turned into different head shapes, and finally a spirits summoned to show its face on the liquid surface.

While deforming the free surface by conventional mesh deformation method may ruin the volume preservation and dynamical nature of the fluid, our method easily produces realistic results, adjustable at interactive rates, taking only several seconds per frame in the preview stage.

The most costly step in our online editing stage (Algorithm 1) is the optimization. While it involves only a handful of parameters, it is non-linear and the computation for the derivatives may need a large number of particles. Although we cannot provide a theoretical complexity analysis and guarantee on its convergence, the optimization converges to a satisfactory solution within 100 iterations with the gradient norm less than $1e^{-3}$, taking only seconds on our complex scenes as shown in Table 1. Even with a large number of sketch strokes and point dragging metaphors, all our tests required less than one minute. However, if large detailed mesh patches are used, as in the spirits example, the optimization is not localized, and usually takes several minutes for a mesh with 1500 vertices. A commodity desktop computer is enough for editing a fluid scene with approximately half a million particles in a moderate grid size. The offline part also runs on the same machine, and it takes 2 min/frame on average to get the final guided detail-enriched animation.

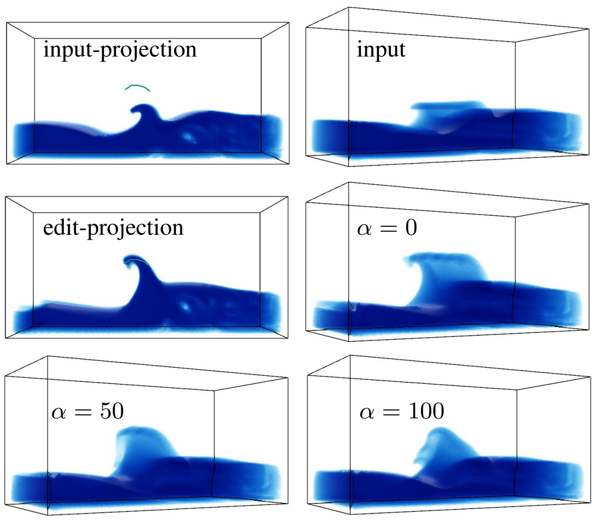

Figure 15: *The user can control the shape of the fluid by a sketch on the projection plane (the sub-figures of input-projection and edit-projection). The influence range in depth is adjusted through α. As shown in the sub-figures, a larger α leads to a smaller patch width.*

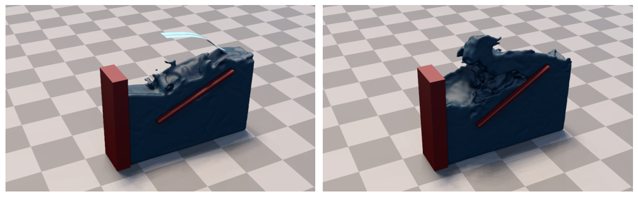

Figure 16: *A single sketch can prototype a plausible splash at the given time. Left: sketch on the input. Right: deformed key frame.*

8 Conclusion

We propose an interactive liquid control method, enabling fast revision and prototyping of liquid animation through intuitive sketch strokes, direct dragging and sub-mesh constraints. Our method provides realtime preview of the edited effects by propagating the editing in a local subsequence in a way that respects the motion of the simulated flow. Lastly, our method performs an offline guided simulation to retain physical details and respect edited motions. The

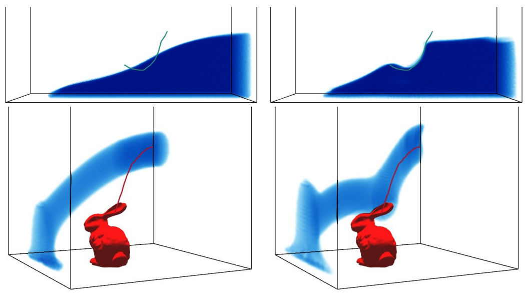

Figure 14: *Locality in the depth direction. Top: the sketch affects all the particles along the line of sight. Bottom: the sketch affects only the closest feature of the liquid body.*

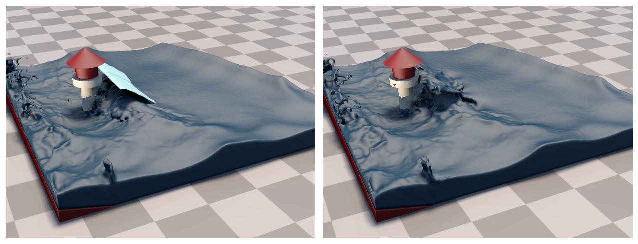

Figure 17: *By sketching on the projection plane, the wave is raised to hit the lighthouse. Efficiency of our method enables interactive editing for desired results. Left: sketch on the input. Right: deformed key frame.*

Scene	Grid Size	Input (M)	Downsampled (K)	Num. Iter.	Time Opt.	Time Warp
Washer	240x32x160	1.2	20.1	27	1500ms	120ms
Lighthouse	320x320x64	4.48	72.3	32	2300ms	450ms
Waterfall	240x160x400	0.6	7.1	50	5700ms	30ms
Spirits	200x200x300	NA	NA	65	5min	3min

Table 1: *Performance statistics: grid size, number of particles (per frame) in the input data and downsampled data (particle number reduces into about 1/64, and the grid resolution reduces to 1/4 in each dimension). The timing is measured on a PC with i7 920 CPU, GTX 560Ti GPU, 16G memory.*

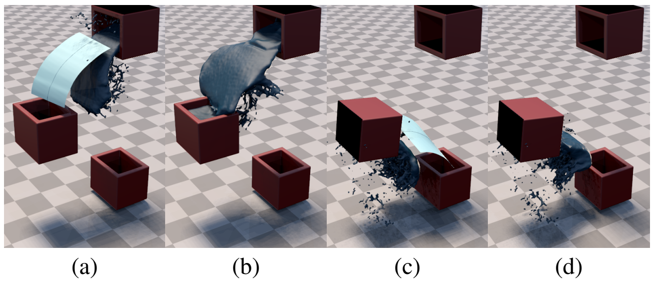

Figure 18: *Applying two consecutive editing passes, water is pulled into lower boxes, changing the semantics of the animation. (a), (c): sketch on the input. (b), (d): deformed key frame.*

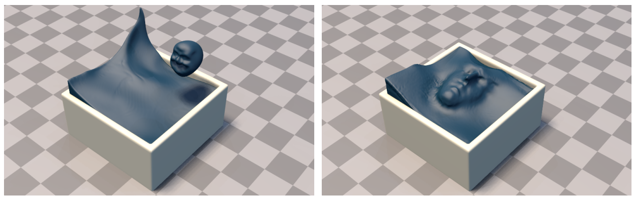

Figure 19: *Spirits in water. Applying mesh patches, we can craft fine details on the free surface and achieve natural fluid motion at the same time.*

interactive performance is realized by a few key components, including the locality of our partial editing metaphors, the geometric nature of our deformer based on the reduced windforce field, and the efficiency of our optimization scheme enabled by the reverse deformation of sketch sample points in our objective function. Our proposed pipeline supports interactive prototyping of a liquid simulation, which can then be used to guide the generation of the detailed high resolution liquid animation.

Limitations and Future Work The major limitation of this work is that we have to sequentially edit the animation, because the edited frame serves as the initial condition for the subesquent simulaiton. An interesting future work is to develop a method to restore the animation by additional control force in the frames following edited keyframes to avoid this causality. Finally, as with all other fluid control methods, artifacts may come from the fictitious force to match the control. Although it may already be visually plausible especially with only moderate modification, better control methods are worth exploring. For efficiency, we only focus on the shape control in this paper, leaving other feasible and useful controls such as velocity control as future work.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was partially supported by NSFC (No. 61170139, No. 61210007), China 973 Program (No. 2009CB320801) and the Fundamental Research Funds for the Central Universities (No. 2013FZA5015). Yiying Tong was supported by NSF (CMMI-1250261 and IIS-0953096). Changxi Zheng was supported by Columbia University junior faculty startup fund.

References

Adams, B., Pauly, M., Keiser, R., and Guibas, L. 2007. Adaptively sampled particle fluids. *ACM Transactions on Graphics 26*, 3, 48:1–48:7.

Ando, R., and Tsuruno, R. 2011. A particle-based method for preserving fluid sheets. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 7–16.

Angelidis, A., and Singh, K. 2007. Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 129–140.

Angelidis, A., Neyret, F., Singh, K., and Nowrouzezahrai, D. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 25–32.

Barbič, J., Sin, F., and Grinspun, E. 2012. Interactive editing of deformable simulations. *ACM Transactions on Graphics 31*, 4, 70:1–70:8.

Barsky, B. A., and Beatty, J. C. 1983. Local control of bias and tension in beta-splines. *SIGGRAPH Comput. Graph. 17*, 3, 193–218.

Barzel, R., Hughes, J. F., and Wood, D. N. 1996. Plausible motion simulation for computer graphics animation. In *Proceedings of the Eurographics workshop on Computer Animation and simulation '96*, Springer-Verlag New York, Inc., 183–197.

Botsch, M., and Kobbelt, L. 2005. Real-time shape editing using radial basis functions. *Comput. Graph. Forum 24*, 3, 611–621.

Botsch, M., and Sorkine, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1, 213–230.

Bridson, R., Houriham, J., and Nordenstam, M. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics 26*, 3, 46.

Chenney, S., and Forsyth, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. *SIGGRAPH Comput. Graph.*, 219–228.

Cohen, M. F. 1992. Interactive spacetime control for animation. *SIGGRAPH Comput. Graph. 26*, 2, 293–302.

Eitz, M., Sorkine, O., and Alexa, M. 2007. Sketch based image deformation. In *Proceedings of Vision, Modeling and Visualization (VMV)*, 135–142.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics 21*, 3, 736–744.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. *ACM Transactions on Graphics 23*, 3, 441–448.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models Image Process. 58*, 5, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *Proceedings of the Conference on Computer Graphics International*.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. *SIGGRAPH Comput. Graph.*, 181–188.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Proceedings of the symposium on Interactive 3D graphics*, 139–ff.

HUANG, J., TONG, Y., ZHOU, K., BAO, H., AND DESBRUN, M. 2011. Interactive shape interpolation through controllable dynamic deformation. *IEEE Transactions on Visualization and Computer Graphics 17*, 7, 983–992.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 409–416.

KAJIYA, J. T., AND VON HERZEN, B. P. 1984. Ray tracing volume densities. *SIGGRAPH Comput. Graph. 18*, 3, 165–174.

KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. *SIGGRAPH Comput. Graph. 24*, 4, 49–57.

KHO, Y., AND GARLAND, M. 2005. Sketching mesh deformations. In *Proceedings of the symposium on Interactive 3D graphics and games*, 147–154.

KIM, T., THÜREY, N., JAMES, D., AND GROSS, M. 2008. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics 27*, 3, 50.

KIRCHER, S., AND GARLAND, M. 2006. Editing arbitrarily deforming surface animations. *ACM Transactions on Graphics 25*, 3, 1098–1107.

LEE, Y., ZITNICK, C., AND COHEN, M. 2011. Shadowdraw: real-time user guidance for freehand drawing. *ACM Transactions on Graphics 30*, 4, 27.

LI, S., HUANG, J., DESBRUN, M., AND JIN, X. 2013. Interactive elastic motion editing through spacetime position constraints. *Computer Animation and Virtual Worlds 24*, 3-4, 409–417.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Transactions On Graphics 23*, 3, 449–456.

MEYER, M., DESBRUN, M., SCHRÖDER, P., BARR, A. H., ET AL. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics 3*, 2, 52–58.

NARAIN, R., SEWALL, J., CARLSON, M., AND LIN, M. C. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Transactions on Graphics 27*, 5, 166:1–166:8.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics 24*, 3, 1142–1147.

NIELSEN, M., AND BRIDSON, R. 2011. Guide shapes for high resolution naturalistic liquid simulation. *ACM Transactions on Graphics 30*, 4, 83.

OLSEN, L., SAMAVATI, F., SOUSA, M., AND JORGE, J. 2009. Sketch-based modeling: A survey. *Computers & Graphics 33*, 1, 85–103.

PASKO, A., ADZHIEV, V., SOURIN, A., AND SAVCHENKO, V. 1995. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer 11*, 8, 429–446.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. *SIGGRAPH Comput. Graph.*, 209–217.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 2007. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3 ed. Cambridge University Press.

RAVEENDRAN, K., THUEREY, N., WOJTAN, C., AND TURK, G. 2012. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 255–264.

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

SHI, L., AND YU, Y. 2005. Taming liquids for rapidly changing targets. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 229–236.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H. 2004. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry Processing*, 175–184.

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

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics 22*, 3, 716–723.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics 25*, 3, 826–834.

VAN DER LAAN, W., GREEN, S., AND SAINZ, M. 2009. Screen space fluid rendering with curvature flow. In *Proceedings of the symposium on Interactive 3D graphics and games*, 91–98.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H. 2006. Vector field based shape deformations. *ACM Transactions on Graphics 25*, 3, 1118–1125.

WOJTAN, C., MUCHA, P. J., AND TURK, G. 2006. Keyframe control of complex particle systems using the adjoint method. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 15–23.

YUAN, Z., CHEN, F., AND ZHAO, Y. 2011. Pattern-guided smoke animation with lagrangian coherent structure. *ACM Transactions on Graphics 30*, 6, 136.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Transactions on Graphics 24*, 3, 965–972.

ZIMMERMANN, J., NEALEN, A., AND ALEXA, M. 2008. Sketching contours. *Computers & Graphics 32*, 5, 486–499.