DOI: 10.1111/cgf.13856
Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

A Rigging-Skinning Scheme to Control Fluid Simulation

Jia-Ming Lu[1], Xiao-Song Chen[1], Xiao Yan[1], Chen-Feng Li[2], Ming Lin[3] and Shi-Min Hu[1]

[1]BNRist, Department of Computer Science and Technology, Tsinghua University, China
[2]College of Engineering, Swansea University, UK
[3]Department of Computer Science, University of Maryland, College Park, USA

Abstract

Inspired by skeletal animation, a novel rigging-skinning flow control scheme is proposed to animate fluids intuitively and efficiently. The new animation pipeline creates fluid animation via two steps: fluid rigging and fluid skinning. The fluid rig is defined by a point cloud with rigid-body movement and incompressible deformation, whose time series can be intuitively specified by a rigid body motion and a constrained free-form deformation, respectively. The fluid skin generates plausible fluid flows by virtually fluidizing the point-cloud fluid rig with adjustable zero- and first-order flow features and at fixed computational cost. Fluid rigging allows the animator to conveniently specify the desired low-frequency flow motion through intuitive manipulations of a point cloud, while fluid skinning truthfully and efficiently converts the motion specified on the fluid rig into plausible flows of the animation fluid, with adjustable fine-scale effects. Besides being intuitive, the rigging-skinning scheme for fluid animation is robust and highly efficient, avoiding completely iterative trials or time-consuming nonlinear optimization. It is also versatile, supporting both particle- and grid- based fluid solvers. A series of examples including liquid, gas and mixed scenes are presented to demonstrate the performance of the new animation pipeline.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

Extensive research in fluid simulation has been carried out to develop fast and plausible simulators for liquid and gas, to improve the simulation capacity for viscous fluid, bubbles and multiphase flow etc., and to capture dynamic interactions between fluids and the environment consisted of rigid, deformable and granular objects. Fluid simulation is now routinely used to create visual effects (VFX) for television and film productions. Despite these great successes, it remains a major technical challenge to intuitively and efficiently animate fluid flows. Due to the flexible and chaotic nature of fluid flow, it can be extremely difficult for animators to specify suitable simulation parameters to achieve the desired flow motion. The current design practice of fluid animation follows largely a rule-of-thumb approach, relies on tedious trial-and-error adjustments, and is particularly painful when working with large scenes. In a VFX pipeline, the trial-and-error iterations of fluid simulation are often the worst bottleneck that limits the quality of visual effects.

The importance of controlling and editing fluid simulation has long been recognized by the graphics community. Due to the complex and unpredictable nature of fluid flows, most previous methods have followed a keyframe-animation strategy in order to reduce the human error of specifying elusive flow motions. Specifically, a small number of keyframes are defined first, and the controlling-editing methods seek external control forces to drive and generate a continuous fluid flow that matches the given keyframes at certain time instances. The difference between the current and target states of fluid flow at each time step is computed in [FL04, SY05, RTWT12] and an additional force dependent on the difference was applied to the fluid. The conflict between the extra control force and the conservation laws of fluid flow causes mismatches to the target keyframes and stability issues to the fluid simulation. As a result, cumbersome scene-dependent parameter adjustments are often required to achieve better results. The control forces are solved as a space-time optimization problem in [TMPS03, MTPS04, PM17], where the objective function is typically defined by combing an error term and a regularization term: the error term measures the difference between the simulation and the target, while the regularization term encourages the conservation laws (e.g. divergence-free constraint) and improves the analytic properties of the objective function (e.g. smoothness and convexity). Depending on the flow regime, the difference to the design target, and the degrees of freedom (DOFs) of the problem, the space-time optimization solution can take many iterations to achieve convergence, which makes the already time-consuming fluid simulation even more expensive to control and edit. For practical animation production, the keyframed animation strategy has several fundamental weaknesses for use in fluid animation: 1) the intermediate flows between keyframes are difficult to control; 2) the overall fluid animation is prone to unnatural and discontinuous

artefacts at keyframes; 3) it is difficult to design complex motions unless a large number of keyframes are specified, greatly increasing the labour work and risk of human errors.

Less common methods with other controlling-editing inputs have also been developed for fluid animation. An embedded controller with a high-level user control of fluid parameters was proposed in [FM97]. The fluid flow was directly driven using individual control particles in [REN*04, TKPR09]. The fluid was guided with low-resolution flows in [NCZ*09, NB11, YCZ11]. There are two common challenges in non-keyframed animation approaches: 1) it is extremely difficult, if not impossible, to specify and edit a large amount of unorganized physical quantities (or control particles); 2) the continuous and somewhat arbitrary control inputs can contradict considerably with the physics of fluid flow causing serious artefacts and even breaking the fluid simulation.

To address the above challenges in fluid animation, the aim of this work is to develop a general-purpose and easy-to-use animation framework to intuitively and efficiently animate fluids. This work is inspired by research in skeletal animation, which is a popular approach to animate characters and mechanical objects for a prolonged period of time (typically over 100 frames). As the default standard for virtually all 3D animation systems, skeleton-driven animation bring characters and mechanical objects alive via two steps: rigging and skinning. Rigging refers to the process of setting up the skeleton in a model, including linking the bones in a hierarchy, setting constraints on the joints, and defining control modes to drive the joints, thereby the motion and deformation of the animated object. Skinning refers to the process of attaching the actual 3D surface mesh to the 3D rig so that the rig will influence the mesh vertices to move and deform the 3D model accordingly. Being intuitive, efficient, robust, and versatile, the key idea of skeletal animation is to split the animation model into the rig that can more intuitively control the model and the skin that can plausibly move and deform the model under the rig control.

We borrow these fundamental concepts from skeletal animation to intuitively and efficiently animate fluids through two integrated steps, fluid rigging and skinning, in an unified simulation system:

- The **fluid rigging** redefines the "skeleton of fluid" in a novel form of a point cloud with rigid-body movement and incompressible deformation. The point-cloud fluid rig can be intuitively specified by the animator via a rigid body motion and a constrained free-form deformation, without concerns regarding the elusive flow features.
- The **fluid skinning** attaches "fluid skin" to the fluid rig and generates plausible fluid flows under the control of the rig. Specifically, at a fixed computational cost, the fluid skin is formed by fluidizing the incompressible point-cloud fluid rig with adjustable zero- and first-order flow features.
- Our control framework for fluid animation is robust and versatile, supporting both particle- and grid-based fluid solvers, and it works for both liquid and gas animations.

2. Related Work

The earliest research work of controlling and editing fluid simulation can be traced back to two decades ago, soon after the introduction of fluid simulation in the graphics community. However, compared to the rapid progress of fluid simulators, controlling and editing of fluids have progressed slowly with a relatively little literature. As summarized in Section 1, the majority of the most relevant previous works have all followed the keyframed animation strategy. In this section, the related literature will be examined from a different perspective to reveal more technical details and to place the current work in a fully illuminated research context.

The concept of fluid control was proposed by [FM97], where an embedded controller was used to control fluid simulation by setting the fluid property, internal or external pressure, flow velocity, boundary condition, etc. The target shape was tracked with the particle level set method in [REN*04], and a soft control was achieved by tuning the particle-generated velocity and adding it to the original velocity field. A driving force was applied to control smoke simulation in [FL04], where a gathering force was introduced to avoid smoke diffusion. [ANSN06] used high level tools, such as animated current curves, attractors and tornadoes to help non-expert users to control smoke animation. Based on the difference between the simulation and target shapes, [SY05] generated a divergence-free velocity field to compensate the original flow velocity. Based on LBM (lattice Boltzmann method), [TKPR09] controlled the fluid flow with an attraction force dependent on the distance between control and fluid particles and a velocity force proportional to the velocity difference. To reduce artificial viscosity effects caused by the control forces, the velocity field was processed with a low-pass filter, so that the low-frequency component was used to compute the velocity force while the high-frequency component was amplified to boost turbulence-like features. A potential energy was proposed by [HK10] to control the shape of fluid, where a potential field was built from the target shape and subsequently used to generate external forces for fluid control. In the aforementioned methods, the fluid control is achieved either by adding control forces or by making direct adjustment to the velocity field, both of which are computed proportional to the difference between the original simulation and the target. These force-velocity intervention schemes can be effective for achieving artificial shapes. But due to the intrusive intervention to the fluid flow, they are prone to unnatural artefacts and stability issues, hence often require post-processing operations to recover flow features. A two-layer approach was described in [MM13], where a bulk velocity drives a particle system towards a target distribution and a vortex particle simulation adds extra fluid motion. Control particles were used in [ZYWL15] to drive the fluid to match the target, where density constraint, spring constraint and velocity constraint were solved under the position-based simulation framework.

To overcome the drawbacks in the direct force-velocity intervention schemes, optimization methods have been employed to automatically determine the force-velocity adjustments. The reference [TMPS03] proposed an objective function consisting of two parts: an error term that measures the difference between current and target shapes and a regularization term that suppresses the intervention forces. an adjoint method was proposed in [MTPS04] to compute the gradient of the objective function, which reduced the cost of iterations in the optimization process. A local optimization approach was proposed in [PHT*13], where the shape and trajectory of liquid flow were controlled by user-specified shape and

path. A blending method was proposed in [RWTT14] to generate a new liquid animation from two or more simulated scenarios. The reference [PM17] used ADMM (alternating direction method of multiplier) to solve a space-time optimization problem for smoke control, which improved the accuracy and efficiency of the optimization process. Compared with the direct force-velocity intervention schemes, these optimization based methods are generally more accurate in achieving the keyframe targets, but they are less intuitive and can suffer from efficiency and convergence issues, depending on the flow regime, the control target and the size of the optimization problem.

Another theme of relevant research is flow guidance and turbulence enhancement. As tuning the simulation parameter for a low-resolution simulation is much faster than that for a high-resolution one, it is desirable to control the costly high-resolution fluid simulation with a pre-computed low-resolution result. For example, [NCZ*09] upsampled a low-resolution simulation and added it to the high-resolution velocity field at each time step. An efficient time-dependent guiding scheme was proposed for smoke simulation [NC10] to improve artistic effects and high-frequency features. The internal guide shape from the low-resolution simulation was extracted in [NB11] and the high-resolution simulation was only performed for the outer shell to add fine-scale details and save cost. The reference [RWTT14] combined the results from different existing fluid simulations. A fast Primal-Dual method was adopted in [IEGT17] to fluid guiding and explicit control was achieved over both large-scale motions and small-scale details. The advected radial basis function was proposed in [PCS04] for modeling and editing flows with an Eulerian approach. Other methods, such as [KTJG08, RLL*13, NSCL08, MBT*15], added fine-scale turbulence effect to the simulation.

Some fluid control techniques have also been used in the animation industry. The reference [WR03] introduced some techniques to interact fluid simulation with other objects; [WH04] created a monster character with fluid simulation; [NSG*17] allowed feasible interaction with the water and wakes near a ship using a locally guided water simulation; [SD16] proposed an animation technique inspired by clay sculpting for fluid animations using interactive direct manipulation of the simulated fluid; [FSN17, SS17] presented water as a character in Disney's film *Moana*.

3. Controlling and Editing Fluid Animation

Unlike most previous approaches where the keyframe animation has been adopted, our fluid control and editing pipeline is inspired by skeletal animation consisted of two integrated steps: rigging and skinning. Rigging is the process of setting the "control skeleton" for the model, including bones, joints and associated constraints and control modes, while skinning is the process of binding the actual 3D mesh to the rig such that the rig can move and deform the model accordingly. The rig allows the animator to intuitively control the model, while the skin makes the rig-controlled model behave like a character (by moving the mesh vertices of the actual model accordingly).

As fluids cannot resist shear force, the rigid rig in character animation would not work for fluid animation. A natural extension is to develop a new form of *deformable rigs*. The fluid rig needs a geometric representation, which should be commonly available, compatible with its motion modes and suitable for the subsequent fluid skinning. Standard geometric representations include NURBS, triangular mesh, point cloud and regular grid etc., among which the point cloud model is most robust and versatile in coping with flexible deformation and topological changes. Hence, we define the 'fluid rig' as a point cloud. Unlike humanoid characters or mechanical objects, fluid flows are fully governed by physical laws, i.e. mass and momentum conservation. In other words, a target motion (or shape sequence) that conflicts with physical laws can never be truthfully achieved on a fluid flow. The momentum conservation depends largely on unknown (often invisible) boundary and loading conditions, and hence it is less intuitive and less restrictive for the fluid control task. However, the mass conservation, i.e. the in incompressibility condition, is not affected by any invisible environmental parameters, and therefore it has a much more intuitive impact on the visual perspective of fluid flow. The incompressibility condition should be respected by the fluid rig, because it is impossible for an inherently incompressible fluid to achieve a compressible motion specified by a fluid rig. Recognizing this intrinsic constraint, we define the fluid rig as a point cloud that undergoes rigid-body movement and incompressible deformation, both of which can be readily specified and manipulated by the animator.

Once the fluid rigging is specified by the animator, the task for the fluid skinning is to generate a fluid skin that respects the given fluid rigging and represents plausible fluid flows. Our approach is inspired by the fluidized bed in chemical engineering, where a pressurized fluid (gas or liquid) passes through a solid particulate substance to form a solid-fluid mixture that behaves like a fluid. We create a *virtual fluidization* process to let the animation fluid interact with the incompressible point-cloud fluid rig. A dedicated mixing law is introduced so that the motion of the fluid rig is accurately transferred to the animation fluid, while both the fluid physics and the user-specified fluid rigging are respected. For practical use, it is crucial to avoid iterative operations in the virtual fluidization, so that the fluid skinning is robust and efficient with fixed computational cost. Another key issue for fluid skinning is to provide the animator with freedom to adjust flow features of the fluid skin.

Throughout the development of the rigging-skinning framework for fluid animation, we have been focusing on intuition, efficiency, robustness and versatility. These research goals have affected our choices of models and algorithms in fluid rigging and fluid skinning, and are explained in detail in the following subsections. Section 3.1 and Section 3.2 explain the formulation details for fluid rigging and fluid skinning, respectively. Section 3.3 summarizes the overall skeletal-animation pipeline for fluids.

3.1. Fluid Rigging

The fluid rigging is explained in three parts. Section 3.1.1 explains the point-cloud geometric representation of fluid rig, while Section 3.1.2 and Section 3.1.3 explain how the motion of the fluid rig is specified and controlled.

3.1.1. Point-cloud Fluid Rig

As shown in Fig. 1, the geometry of a fluid rig is represented by a point cloud, which can be formed via standard methods, e.g. through direct construction or sampling a given geometric model. For simplicity, we refer to the points in a fluid rig as rig points. No special criterion is needed to form the point-cloud fluid rig, but the usual geometric requirement applies, i.e. the distribution density of rig points should be consistent with the local geometric features to avoid waste. If the fluid rig is built by sampling a geometric entity, a body-centered lattice point generator can be used for homogeneity and a Poisson sampler can be used for randomness.

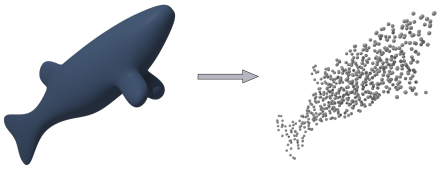

Figure 1: *A point-cloud fluid rig*

3.1.2. Rigid Body Rig Motion

As discussed earlier, the fluid rig should at least satisfy the incompressibility condition, otherwise the user-specified motion can never be truthfully achieved by the intrinsically incompressible fluid flow. The rigid-body movement is a special form of motion that satisfies the incompressibility condition, and is the simplest way to move a fluid rig. As shown in Fig. 2, a rigid body motion can be specified to a fluid rig, either through a user-specified function or a manual sketch.

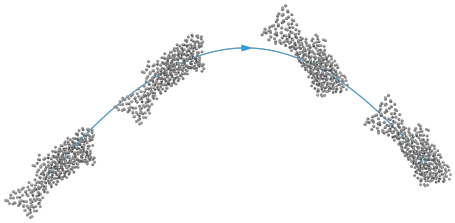

Figure 2: *A fluid rig with a rigid body motion*

3.1.3. Incompressible Rig Deformation

A more general mode of motion a fluid rig could undertake is the incompressible deformation, as illustrated in Fig. 3. Held by all common fluids, the incompressibility condition adds a strict constraint to the flow velocity and has a dominant impact on the visual appearance of fluid flows. It is essential for the fluid rig to satisfy the incompressibility condition so that its control input would cause minimum physical conflicts with the animation fluid, which is arguably the main source of artefacts and stability issues.

There are many different ways to generate incompressible deformation. For convenience and simplicity, we implemented the

Figure 3: *A fluid rig with an incompressible deformation*

vector field based shape deformations (VFSD) [vFTS06], a popular free-form deformation method. The VFSD method supports incompressible moving, stamping, bending and twisting for intuitive free-form deformation. The underlying computation is efficiently realized via four explicit steps:

1. Two simple analytic scalar fields $e(\boldsymbol{x})$ and $f(\boldsymbol{x})$ are first defined, where $\boldsymbol{x}(x,y,z)$ denotes a point in space. Given a normalized direction vector $\boldsymbol{v}$ and a center point $\boldsymbol{c}$, the translation field is defined as:

$$e(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x}-\boldsymbol{c})^T, f(\boldsymbol{x}) = \boldsymbol{w}(\boldsymbol{x}-\boldsymbol{c})^T \tag{1}$$

   where $||\boldsymbol{u}|| = ||\boldsymbol{w}|| = 1$ and $\boldsymbol{uv} = \boldsymbol{vw} = \boldsymbol{wu} = 0$. Given a normalized rotational axis direction $\boldsymbol{a}$ and a center point $\boldsymbol{c}$, the rotation field is defined as:

$$e(\boldsymbol{x}) = \boldsymbol{a}(\boldsymbol{x}-\boldsymbol{c})^T, f(\boldsymbol{x}) = (\boldsymbol{a} \times (\boldsymbol{x}-\boldsymbol{c})^T)^2 \tag{2}$$

2. The deformation regions are defined by a scalar field $r(\boldsymbol{x})$ with two constant thresholds $r_i < r_o$. If a point $\boldsymbol{x}$ is in the inner region (i.e. $r(\boldsymbol{x}) < r_i$), $\boldsymbol{x}$ undergoes a full deformation; if $\boldsymbol{x}$ is in the outer region (i.e. $r(\boldsymbol{x}) \geq r_o$), $\boldsymbol{x}$ remains undeformed; and for $\boldsymbol{x}$ in the intermediate region (i.e. $r_i \leq r(\boldsymbol{x}) < r_o$), the deformation of $\boldsymbol{x}$ is obtained by a blending operation. The blending function $b(r(\boldsymbol{x}))$ is defined as:

$$b(r) = \sum_{j=0}^{4} w_j B_j^4\left(\frac{r-r_i}{r_o-r_i}\right) \tag{3}$$

   where B_j^4 denotes the well-known Bernstein polynomials and $(w_0, w_1, w_2, w_3, w_4) = (0,0,0,1,1)$.
3. Based on analytic scalar fields $e(\boldsymbol{x})$ and $f(\boldsymbol{x})$ and the blending function $b(r(\boldsymbol{x}))$, two blended scalar fields $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ can be constructed as:

$$p(\boldsymbol{x}) = \begin{cases} e(\boldsymbol{x}) & \text{if } r(\boldsymbol{x}) < r_i \\ (1-b)\cdot e(\boldsymbol{x}) & \text{if } r_i \leq r(\boldsymbol{x}) < r_o \\ 0 & \text{if } r_o \leq r(\boldsymbol{x}) \end{cases} \tag{4}$$

$$q(\boldsymbol{x}) = \begin{cases} +f(\boldsymbol{x}) & \text{if } r(\boldsymbol{x}) < r_i \\ (1-b)\cdot f(\boldsymbol{x}) & \text{if } r_i \leq r(\boldsymbol{x}) < r_o \\ 0 & \text{if } r_o \leq r(\boldsymbol{x}) \end{cases} \tag{5}$$

4. Using the blended scalar fields $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, a divergence-free vector field $\mathbf{v}$ can be constructed for incompressible deformation:

$$\mathbf{v}(\boldsymbol{x}) = \nabla p(\boldsymbol{x}) \times \nabla q(\boldsymbol{x}) \tag{6}$$

3.2. Fluid Skinning

The fluid skinning converts the user-specified motion of the fluid rig into plausible flow motions of the animation fluid. This is achieved via two steps: rig fluidization and linear flow skinning. In the rig fluidization step, the fluid rig defines a continuous force field distributed across the rig geometry, and it mobilizes the animation fluid to follow consistently the move of the rig. At fixed computational cost, the linear flow skinning step generates a fluid skin with adjustable zero-order (i.e. velocity) and first-order (i.e. velocity gradient) flow features.

3.2.1. Rig Fluidization

We create a virtual fluidization process and let the animation fluid pass through the fluid rig and interact with the cloud of rig points. In the virtual fluidization, each rig point exerts a virtual drag force due to the mismatch between the flow motion of the animation fluid and the user-specified motion of the fluid rig. The virtual drag force exerted by the i-th rig point p_i is:

$$\boldsymbol{F}_i = -D_V \Delta \boldsymbol{v}_i - D_T \Delta \boldsymbol{v}_i ||\Delta \boldsymbol{v}_i|| - D_I \Delta \dot{\boldsymbol{v}}_i = \boldsymbol{F}_i^V + \boldsymbol{F}_i^T + \boldsymbol{F}_i^I \quad (7)$$

where $\Delta \boldsymbol{v}_i$ is the relative velocity between the rig point p_i and the local animation fluid, $\Delta \dot{\boldsymbol{v}}_i$ is the relative acceleration, $\boldsymbol{F}_i^V = -D_V \Delta \boldsymbol{v}_i$ represents the virtual drag force due to the viscous effect, $\boldsymbol{F}_i^T = -D_T \Delta \boldsymbol{v}_i ||\Delta \boldsymbol{v}_i||$ represents the virtual drag force due to the turbulent effect, $\boldsymbol{F}_i^I = -D_I \Delta \dot{\boldsymbol{v}}_i$ represents the virtual drag force due to the inertial effect, D_V denotes the viscous drag coefficient, D_T denotes the turbulent drag coefficient, and D_I denotes the inertial drag coefficient.

By using a standard spatial interpolation, the virtual drag forces exerted by individual rig points define a force field distributed continuously across the rig geometry. The continuous rig force field mobilizes the animation fluid with a low-frequency motion that is consistent with the move of the rig, thanks to its incompressibility condition. Specifically, the rig force at the position $\boldsymbol{x}(x,y,z)$ in the animation fluid is:

$$\boldsymbol{F}(\boldsymbol{x}) = -\frac{\sum_i W(\boldsymbol{x}_i, \boldsymbol{x}) \boldsymbol{F}_i}{\sum_i W(\boldsymbol{x}_i, \boldsymbol{x})} \quad (8)$$

where $\boldsymbol{x}_i$ is the position of p_i, $W(\boldsymbol{x}_i, \boldsymbol{x})$ denotes the interpolation value for $\boldsymbol{x}$ around $\boldsymbol{x}_i$. The summation is performed for all rig points p_i within the neighbourhood of the position $\boldsymbol{x}$ such that $||\boldsymbol{x} - \boldsymbol{x}_i|| \leq R$, where R is the influencing radius of a rig point.

To convert the motion of the fluid rig into a plausible fluid flow, the animation fluid is mobilized by the continuous rig force $\boldsymbol{F}(\boldsymbol{x})$ as defined in Eqn. 7 and Eqn. 8. In Eqn. 7, the three virtual drag forces have distinct physical/numerical effects, and they are not interchangeable. In Eqn. 8, the rig force at an arbitrary position in the animation fluid is extracted from the continuous force field exerted by the fluid rig, which is independent from the resolution of the specific fluid solver. It should be noted that:

- The incompressibility condition of the fluid rig is essential and critical, without which a truthful control can never be achieved as the specified motion will violate the physics of fluid flow.
- The rig fluidization is independent from the type of fluid solvers and is applicable to both liquid and gas flows.

In previous particle-based control methods, drag force has also been used to guide fluids. In [TKPR09], a velocity term and an attraction force term are combined to add the external control force. The velocity term suppresses the difference between the fluid velocity and the control velocity to make the fluid follow the controlled motion. However, when the control motion has an acceleration, the velocity term is insufficient to control the fluid. Therefore, the additional attraction force term is introduced to attract the fluid towards the control particles. In our experiments, we found the attraction force term is unstable and often causes unwanted side effects. To overcome these problems, an inertial drag force is introduced instead, to directly link the drag force with the acceleration difference. Fig. 4 shows a comparison between our approach and the method in [TKPR09], where the attraction force [TKPR09] causes a lot of undesired vortices near the edge of letter "F".

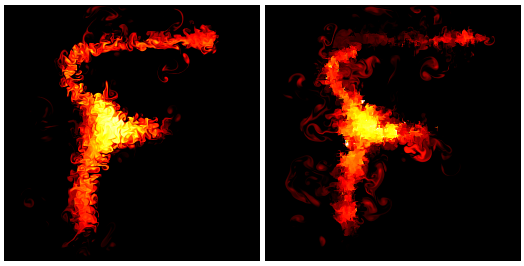

Figure 4: *Comparison between our rig fluidization approach and the method in [TKPR09]. The left figure shows the final shape of letter "F" created by using rig fluidization, and the right figure shows the result by using method proposed in [TKPR09].*

3.2.2. Linear Flow Skinning

Let $\boldsymbol{v}_A$ denote the original velocity of the animation fluid and $\boldsymbol{v}_{Af}$ the velocity of the animation fluid derived from the rig fluidization. As the incompressibility condition is strictly respected by the user-specified motion, the derived velocity field $\boldsymbol{v}_{Af}$ is guaranteed to be a plausible flow field, which have been confirmed in all our examples. However, for practical animation production, it is still desirable to provide the animator with freedom to adjust fine-scale features that are not directly controlled by the fluid rig. This is achieved by the linear flow skinning step.

We consider both zero-order features described directly by the velocity field and first-order features represented by the velocity gradient, and a linear optimization problem is formed to update the velocity field $\boldsymbol{v}_{Af}$ of the animation fluid:

$$\boldsymbol{v}_{As} = \arg\min_{\boldsymbol{v}} \Phi(\boldsymbol{v}_{Af}, \boldsymbol{v}) \quad (9)$$

where $\Phi(\boldsymbol{v}_{Af}, \boldsymbol{v})$ is a quadratic objective function, $\boldsymbol{v}$ denotes the independent variable and $\boldsymbol{v}_{As}$ denotes the velocity of the animation fluid after linear flow skinning. The objective function for linear flow skinning is defined as:

$$\begin{aligned} \Phi(\boldsymbol{v}_{Af}, \boldsymbol{v}) &= \lambda_1 ||\nabla \boldsymbol{v} - \nabla \boldsymbol{v}_1||^2 + \lambda_0 ||\boldsymbol{v} - \boldsymbol{v}_0||^2 \\ &= \Phi_1(\boldsymbol{v}_{Af}, \boldsymbol{v}) + \Phi_0(\boldsymbol{v}_{Af}, \boldsymbol{v}) \end{aligned} \quad (10)$$

where $\Phi_1(\boldsymbol{v}_{Af}, \boldsymbol{v}) = \lambda_1 ||\nabla \boldsymbol{v} - \nabla \boldsymbol{v}_1||^2$ encourages the first-order

flow features specified by $\nabla \boldsymbol{v}_1$ and $\Phi_0(\boldsymbol{v}_{Af}, \boldsymbol{v}) = \lambda_0 ||\boldsymbol{v} - \boldsymbol{v}_0||^2$ encourages the zero-order flow features specified by $\boldsymbol{v}_0$.

- The term $\Phi_1(\boldsymbol{v}_{Af}, \boldsymbol{v})$ is responsible for the adjustment of first-order flow features. This is the recommended route to add flow features, as it spreads locally the adjustments based on the velocity gradient, hence the velocity update is more gentle and stable. The velocity-gradient feature $\nabla \boldsymbol{v}_1$ can be freely specified by the animator based on the practical demand. A straightforward application is to add fine-scale turbulence. We use the curl-noise [BHN07] to generate random potential fields $\boldsymbol{\Psi} = (\Psi_1, \Psi_2, \Psi_3)$ and construct a turbulent velocity field:

$$\boldsymbol{v}_T = \left(\frac{\partial \Psi_3}{\partial y} - \frac{\partial \Psi_2}{\partial z}, \frac{\partial \Psi_1}{\partial z} - \frac{\partial \Psi_3}{\partial x}, \frac{\partial \Psi_2}{\partial x} - \frac{\partial \Psi_1}{\partial y}\right) \tag{11}$$

  The gradient $\nabla \boldsymbol{v}_T$ can be taken as $\nabla \boldsymbol{v}_1$ to add turbulent features. Another application is to encourage the original flow $\boldsymbol{v}_A$ of the animation fluid. This is particularly useful when $\boldsymbol{v}_A$ has been established and the task is to carry out further adjustment using the rig. In this case, it could be necessary to retain some of the original flow features. This can be achieved by simply assigning $\nabla \boldsymbol{v}_A$ to $\nabla \boldsymbol{v}_1$ as the first-order flow feature for the fluid skin. Finally, a more flexible adjustment to first-order flow features can be realized by decomposing and modulating the velocity-gradient flow feature $\nabla \boldsymbol{v}_1$. Specifically, the following tensor decomposition holds

$$\begin{aligned} \nabla \boldsymbol{v}_1 = & \underbrace{\frac{1}{2}(\nabla \boldsymbol{v}_1 - (\nabla \boldsymbol{v}_1)^T)}_{\boldsymbol{R}_1} + \underbrace{\frac{1}{3}(\nabla \cdot \boldsymbol{v}_1)\boldsymbol{I}}_{\boldsymbol{V}_1} \\ & + \underbrace{\left(\frac{1}{2}(\nabla \boldsymbol{v}_1 + (\nabla \boldsymbol{v}_1)^T) - \frac{1}{3}(\nabla \cdot \boldsymbol{v}_1)\boldsymbol{I}\right)}_{\boldsymbol{S}_1} \end{aligned} \tag{12}$$

  where $\boldsymbol{R}_1$ is the spin tensor representing the vorticity of the motion, $\boldsymbol{V}_1$ is rate-of-expansion tensor representing the volume change of the motion, $\boldsymbol{S}_1$ is the rate-of-shear tensor representing the shape change of the motion. Following the above velocity-gradient decomposition, the first-order features of the flow motion (i.e. vorticity, volume change and shape change) can be modulated by purposely mixing the individual motion components as:

$$\nabla \boldsymbol{v}_1 = \alpha_R \boldsymbol{R}_1 + \alpha_V \boldsymbol{V}_1 + \alpha_S \boldsymbol{S}_1 \tag{13}$$

  where $\alpha_R \in (0,1)$, $\alpha_V \in (0,1)$ and $\alpha_S \in (0,1)$ are weight parameters to modulate the vorticity, expansion and shear features, respectively.
- The term $\Phi_0(\boldsymbol{v}_{Af}, \boldsymbol{v})$ is responsible for the adjustment of zero-order flow features, and it also serves as a regularization term to improve the convexity and smoothness of the objective function. Directly operating on the velocity value, the term $\Phi_0(\boldsymbol{v}_{Af}, \boldsymbol{v})$ can cause much more intrusive changes to the animation flow, and care must be taken to avoid artefacts and instability. In all our examples, we only use $\Phi_0(\boldsymbol{v}_{Af}, \boldsymbol{v})$ as a regularization term, where the zero-order flow feature $\boldsymbol{v}_0$ is simply set as $\boldsymbol{v}_{Af}$.

3.3. Rigging-Skinning Pipeline for Fluid Animation

The overall workflow of the proposed skeletal-animation framework for fluids is summarized in Algorithm 1. Similar to the rigging for animating characters, the fluid rigging is done offline by the animator. Unlike character animation, the fluid must be animated without violating its intrinsic physics. This is ensured by enforcing the incompressibility condition on the fluid rig and by driving the fluid flow with a combination of viscous, turbulent and initial forces in the rig fluidization step. For additional flexibility, the linear flow skinning allows such fine-scale features as turbulence, vorticity, shear and expansion to be individually adjusted. The fluid-rigging and fluid-skinning operations are both independent from the type of fluid solvers, and they are applicable to both liquid and gas flows. The rigging-skinning pipeline for fluids is simple, but it works and achieves intuition, efficiency, robustness and versatility in fluid animation.

ALGORITHM 1: Rigging-Skinning Pipeline for Fluids Animation

Fluid Rigging: Animators create and edit the point-cloud fluid rig and specify its motion with rigid body motion and incompressible deformation.
repeat
  Compute gravity force
  Rig Fluidization: Compute a continuous rig force as external force based on the motion specified on the fluid rig
  Update velocity with computed external force
  Solve equations for fluid incompressiblity with modified velocity
  Linear Flow Skinning: The zero-order and first-order flow features are adjusted via linear flow skinning, where turbulence, vorticity, shear and expansion features are individually adjustable.
  Advection with skinned velocity
until *end of simulation*;

4. Examples and Results

We have implemented the proposed rigging-skinning framework on both particle-based and grid-based fluid solvers. All demos are simulated on a PC platform with Intel(R) Xeon(R) E5-2650 v2 CPU and NVIDIA GTX 1080Ti GPU. A total of five demos are presented in this section, including liquid, gas and mixed scenes. The simulation parameters for all five examples are listed in Table 1, while their implementation details are discussed later. In these examples, we will demonstrate the importance of the incompressibility condition for the fluid rig and examine the specific visual effects corresponding to individual skinning terms. A supplementary video is provided for a closer observation of both the fluid rigging and the controlled fluid flow.

During fluid rigging, all rig points are visible and can be modified by the animator, but the distribution density of rig points needs to be considered and determined before editing the motion of the fluid rig. The experimental results show that having too dense rig points will cause the animation fluid to completely follow the rig, losing its own flow details, while having too sparse rig points may not represent the desired shape well. In our experiments, the appropriate particle density is achieved when the spacing between rig

Table 1: *Simulation parameters.*

Case	R	D_V	D_T	D_I	λ_1	λ_0
Waterfall	$8H$	0.0	5.0	0.0	0.01	1.0
Letters "FLUID"	$4H$	100.0	0.0	1.0	0.0001	1.0
Dancing Dolphin	$6H$	100.0	0.0	1.0	0.01	1.0
Twisted Smoke	$8H$	100.0	0.0	1.0	-	-
Waterspout "SIG"	$4H$	200.0	1.0	0.0	-	-

H denotes the particle spacing or grid spacing in fluid simulation.

points is 4-8 times of the simulation resolution (i.e. spacing between fluid particles or the grid size). Hence, measured by DOFs, the fluid rig is typically hundreds of times smaller than the animation fluid. Except for the calculation of the virtual rig force, the rig points do not participate into the fluid simulation, and therefore the size of the fluid rig has little impact on the overall computational cost of fluid animation.

The three virtual drag forces defined in Eqn. 7 have distinct physical/numerical indications as explained in Section 3.2.1. They have different visual effects when applied to rig fluidization, and are not interchangeable. A comparison example is given in Fig. 5, where by using the same fluid rigging the left letter "F" is generated with the viscous drag force and the right letter "F" is generated with the turbulent drag force. It can be observed that the visual appearances are rather different, with the viscous drag force creating a smoother effect and the turbulent drag force creating a chaotic effect. Depending on the VFX requirement, the three drag forces need to be specifically balanced in rig fluidization. In this work, the waterfall example is created with the turbulent and inertial drag forces, the letters "FLUID" example is created with the viscous and turbulent drag forces, while the remaining three examples are created with the viscous drag force. To examine the performance of the proposed skeletal-animation framework for fluids, we tested three fluid solvers: WCSPH [BT07], IISPH [ICS*14] [WBZ*17] and a grid-based smoke solver [FSJ01]. In our implementation, the PCG (preconditioned conjugate gradients) method was used to solve the equation for the grid-based fluid solver, and the CG (conjugate gradients) method was used for the particle-based fluid solvers.

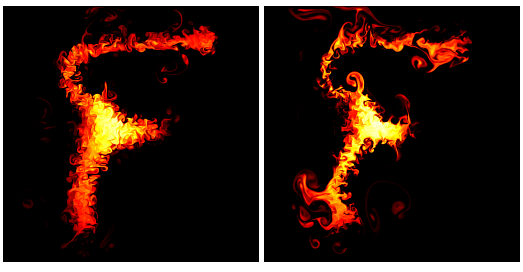

Figure 5: *Comparison between the viscous drag force and the turbulent drag force. The left figure shows the final shape of letter F created by using the viscous drag force, and the right figure shows the result from the turbulent drag force.*

4.1. Waterfall

This test case is set up following the waterfall example in [PHT*13]. The top row in Fig. 6 shows the scene and the animation result from [PHT*13]. Water in the top box is poured into the middle one and then into the bottom one. By specifying a sketch, [PHT*13] guided the trajectory of the waterfall. Similar trajectory guide can also be achieved by our approach, as demonstrated by the bottom row in Fig. 6. In this example, we adopt an explicit particle-based fluid solver WCSPH [BT07] implemented on GPU with 871K SPH (smoothed particle hydrodynamics) particles. The fluid rig is defined as a regularly distributed point cloud in a simple cuboid shape, a path is drawn to indicate the desired trajectory, and the motion of the fluid rig is specified as a rigid-body movement following the path. The corresponding animation result is shown in the bottom row of Fig. 6, where the waterfall is guided to follow the user-specified trajectory.

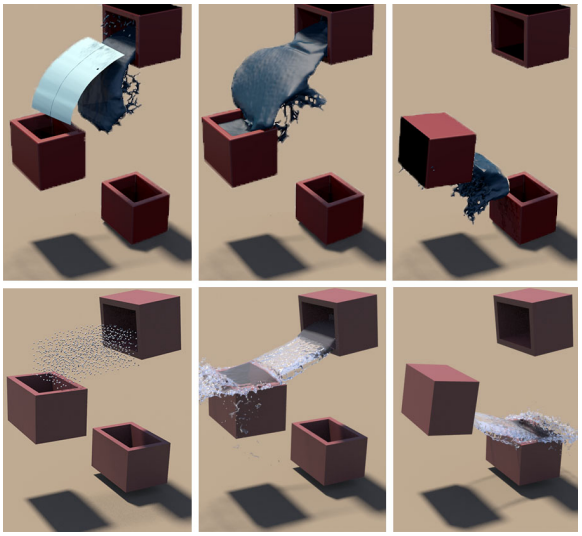

Figure 6: *Comparison of waterfall. The top row shows the animation result of [PHT*13]. The bottom row shows the result from our fluid rigging-skinning method, where trajectory control is simply achieved by specifying a rigid-body movement on the fluid rig.*

To examine the performance of linear flow skinning, three fluid skins are presented in the left, middle and right columns of Fig. 7. The left column shows the animation result without linear flow skinning; the middle column shows the animation generated by using $\nabla \boldsymbol{v}_A$ as the first-order flow feature in linear flow skinning; and the right column shows the animation generated by using a modulated velocity gradient as the first-order flow feature in linear flow skinning. Without linear flow skinning, the waterfall front shown in the left column exhibits some artefacts caused by over rigging. These small-scale artefacts can easily be removed by linear flow skinning without changing the rig, as demonstrated in the middle and right columns. Furthermore, the right column shows enhanced vorticity motions achieved by using a modulated velocity gradient in linear flow skinning.

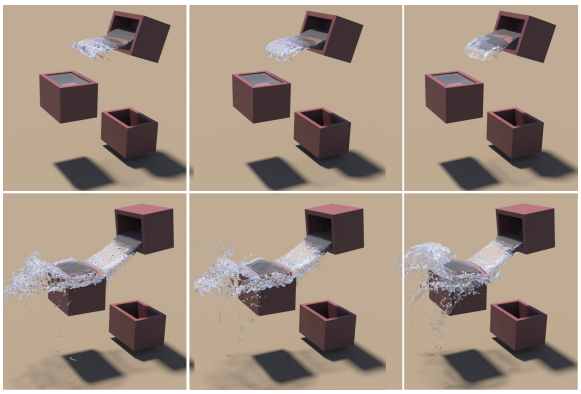

Figure 7: *Different visual effects from linear flow skinning. The left column shows the animation result without linear flow skinning. The middle column shows the result where the gradient of the uncontrolled velocity field is used as the first-order flow feature in linear flow skinning. The right column shows an enhanced vorticity motion, where a modulated velocity gradient is taken as the first-order flow feature for linear flow skinning.*

4.2. Letters "FLUID"

This test case is set up following the letters “FLUID” example in [PM17]. As shown in Fig. 8, smoke is animated to form letters “FLUID”. The first row shows the results from [PM17], which took a keyframe-animation strategy and solve the flow motion as a space-time optimization problem. We adopt a grid-based non-viscous fluid solver [FSJ01] in this example, with a simulation grid of 512×512. The 2nd, 3rd and 4th rows in Fig. 8 show the animation results generated by using our skeletal-animation approach, where different visual effects are created with different linear flow skinning.

Figure 8: *Comparison for letters “FLUID”. The first row shows the result of [PM17]. The 2nd, 3rd and 4th rows show the results from our method, where the 2nd are animated without linear flow skinning and the 3rd and 4th rows are animated with viscous and turbulent linear flow skinning.*

In our fluid rigging-skinning approach, users can easily control the the intermediate motion of the smoke by interactively manipulating the fluid rig. The 1st row in Fig. 9 shows the fluid rigging to animate the letter "F", and the result generated without linear flow skinning is shown in the 2nd rows, respectively. The 3rd and 4th row in Fig. 9 shows a viscous effect and a turbulent generated by modulating the shear or spin motion component during linear flow skinning, and the effects on all five letters are shown in the 3th and 4th row of Fig. 8.

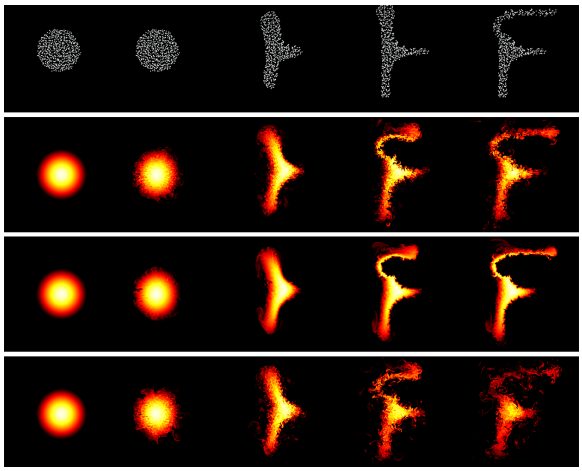

Figure 9: *The effects from different fluid rigs and linear flow skinning. The 1st row shows the fluid rigging to animate the letter “F”. The 2rd row shows the animation generated by using the fluid rigging in row 1. The results shown in row 3 and row 4 are animated and adjusted to give a viscous and turbulent look respectively using linear flow skinning.*

4.3. Dancing Dolphin

An implicit particle-based fluid solver IISPH [ICS*14] is tested in this example, and the animation scene contains about 6.1M SPH particles. Using the fluid rigs as shown in Fig. 1, Fig. 2 and Fig. 3, a water dolphin is formed and animated to jump from the left tank to the right tank, as shown in Fig. 10. As well as jumping between the water tanks, the water dolphin dances by swing its tail, nodding its head and twisting its body. The complex motion undertaken by the water dolphin is difficult to realize with the keyframed animation strategy, but it is straightforward to animate using fairly simple fluid rigging. Effects of different sampling densities for the point rig are shown in this demo. Finer rig points make the fluid follow the dancing motion more closely with less free-flow features, while coarser rig points bring more free-flow features with a compromised dancing motion.

4.4. Twisted Smoke

A grid-based non-viscous fluid solver [FSJ01] is adopted in this example with a simulation grid of $128 \times 128 \times 128$. Three columns of smoke with different colors are twisted 360 degrees. The 1st and 2nd row show the rig and simulation result with a twisting rig

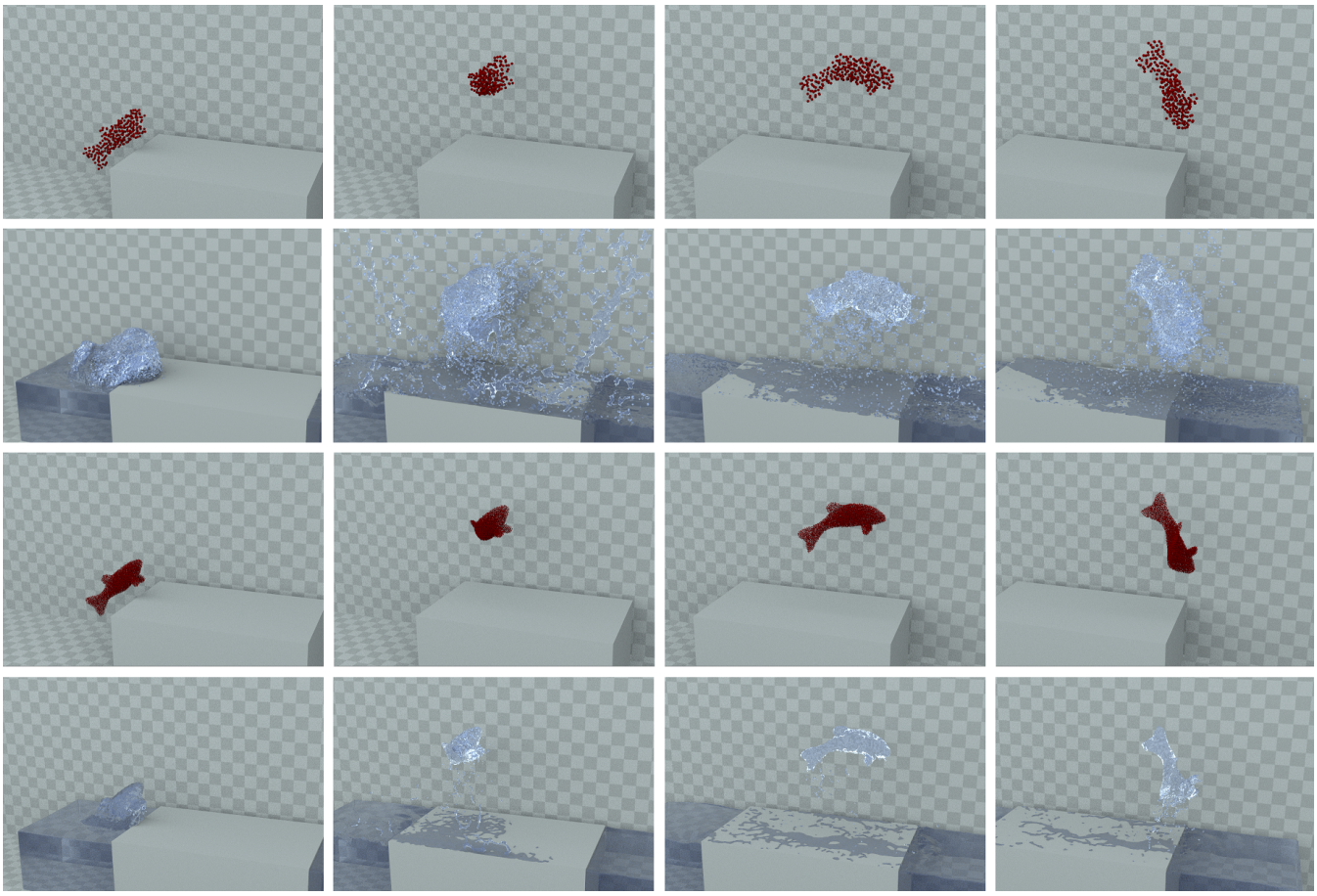

Figure 10: *A dancing dolphin. A water dolphin jumping from the left tank into the right tank, and it swings tail, nods head and twists body before getting into the water again. The 1st and 2nd rows show the result of a coarse point rig. The 3rd and 4th rows show the result of dense point rig. The 1st and 3rd rows show the fluid rig edit by users. The 2nd and 4th rows show the final simulation result.*

generated by the incompressible rig deformation tool. It can be observed that the motion specified by the fluid rig is truthfully transferred to the smoke flow. The 3rd and 4th row show the rig and simulation result with simple matching of several twisting keyframes, which does not satisfy the impressibility condition. We sampled five keyframes from the twisting rig shown in the 1st row and used uniform motion instead of incompressible deformation to generate the rig. It can be observed that the smoke flow fails to follow the motion specified by the fluid rig. Due to the conflict between the compressible rig motion and the incomprehensibility condition of smoke, the resulting smoke flow is rather arbitrary. This example demonstrates the impressibility condition is absolutely essential for any fluid rig.

4.5. Waterspout

This example tests the new fluid rigging-skinning approach on a mixed scene involving both liquid and gas flows. A waterspout animation is generated, where water and vapour are animated together using the same fluid rig to form three waterspouts. The water flow is simulated using an explicit particle-based solver WCSPH [BT07] with 10.9M SPH particles and the vapour flow is simulated using a grid-based smoke solver [FSJ01] with a $256 \times 128 \times 128$ grid. The fluid rigging is formed by a divergence-free tornado field. The fluid rigging and animation result are shown in Fig. 12.

4.6. Performance

In terms of DOFs, the point-cloud fluid rig model is typically hundreds of times smaller than the animation model. The fluid rig can only undertake rigid-body movement and incompressible deformation, which can both be easily specified (either automatically or interactively) with little computational cost. Hence, fluid rigging can easily achieve real-time performance on ordinary computers. Fluid skinning contains two operations: rig fluidization and linear flow skinning. Rig fluidization requires a fluid simulation with added rig forces, so the computational overhead is similar to that of a standard fluid simulation. Linear flow skinning needs to solve a linear equation, whose size is the same as the pressure equation in an implicit fluid solver. But due to the velocity gradient term in linear flow skinning, it may take 4-8 times longer to solve. Therefore, the computational efficiency of fluid skinning is at the same order

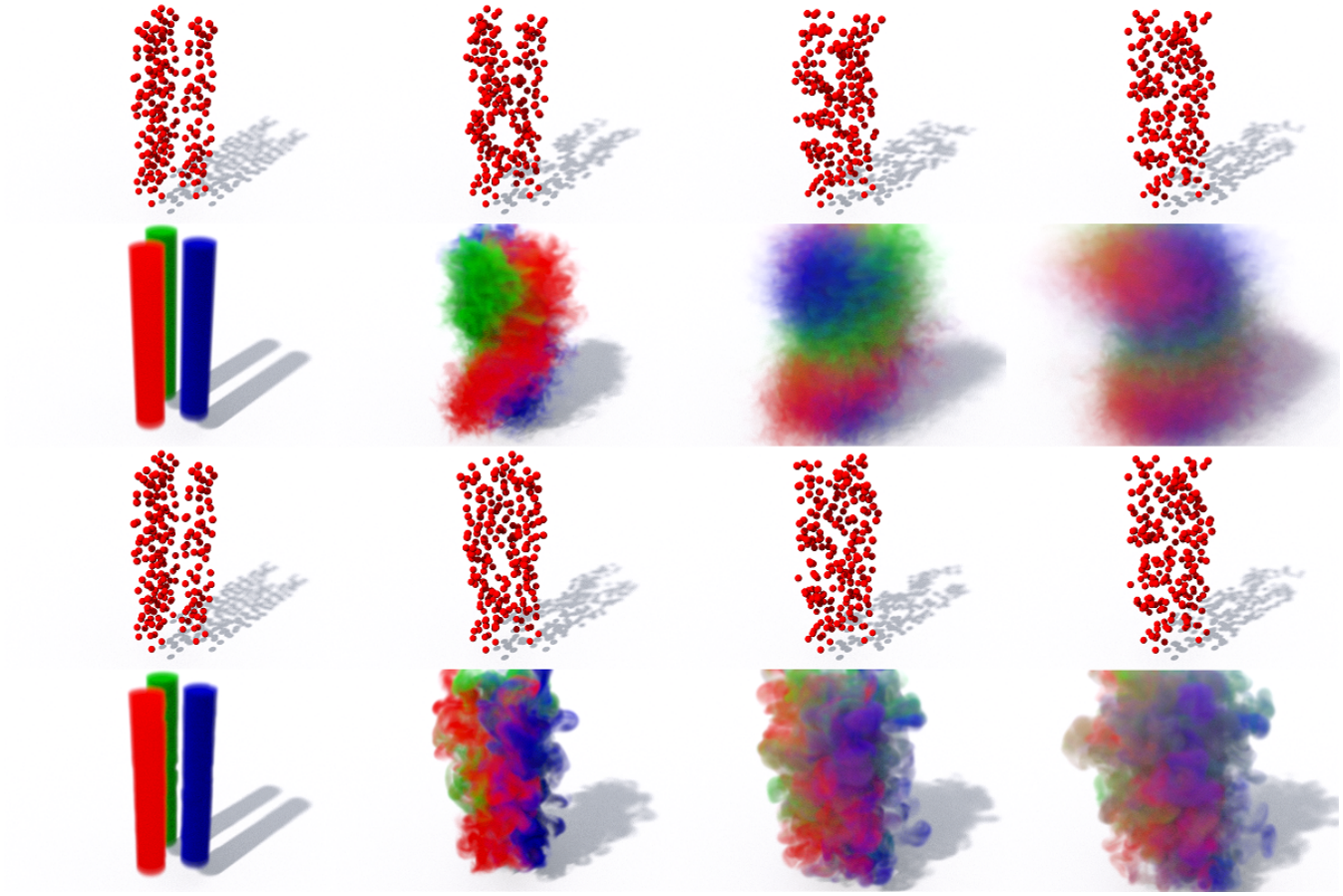

Figure 11: *Twisted smoke. The 1st and 2nd row show the rig and simulation result with a twisting rig generated by our incompressible rig deformation tool. The 3rd and 4th row show the rig and simulation result with simple matching of several twisting keyframes, which does not satisfy the incompressibility condition.*

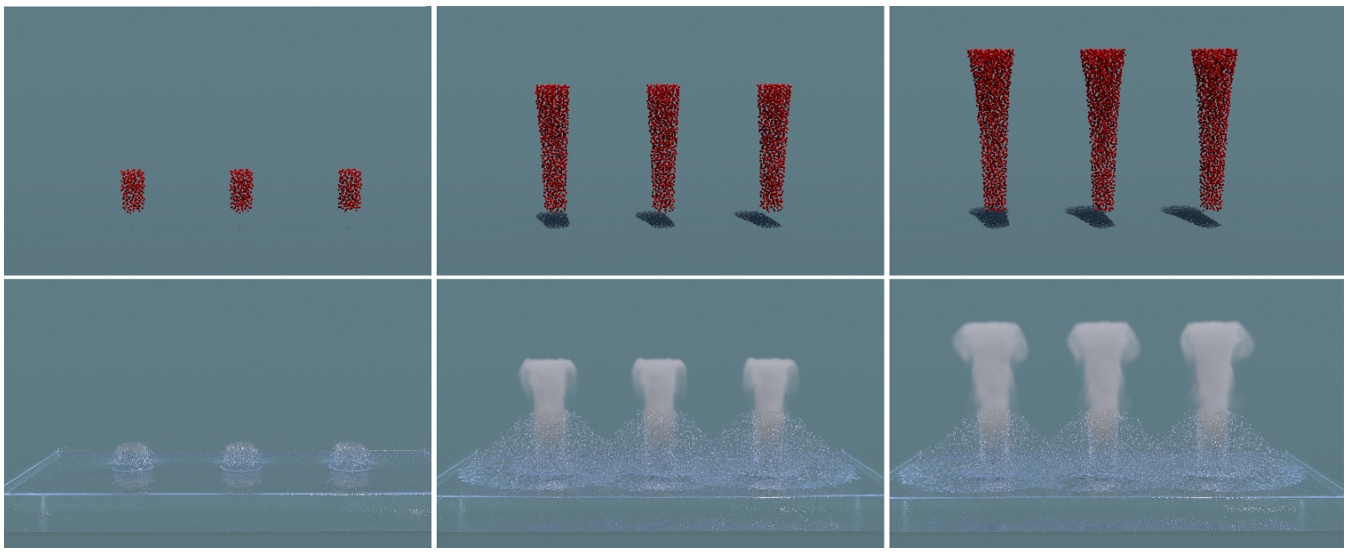

Figure 12: *Waterspout. The 1st row shows the waterspout rig generated by a divergence-free tornado field. The 2nd row shows the final simulation result.*

Table 2: *Performance data.*

Case	Resolution	Rig points	Δt	Time steps	Linear flow skinning	Avg. time (ms/step)	Tot. time
Waterfall	871K particles	21 K	0.00012	41666	N	47	33min
					Y	122	85min
Letters "FLUID"	512×512	2,376	0.01	100	N	1,730	3min
					Y	7,800	14min
Dancing Dolphin	6.1M particles	7.4K	0.001	4000	N	2299	2.5hour
		171	0.0006	6667	Y	9366	17.3 hour
Twisted Smoke	$128 \times 128 \times 128$	225	0.01	100	N	12,000	20min
Waterspout (water)	10.9M particles	16K	0.0018	16666	N	398	110min
Waterspout (vapor)	$256 \times 128 \times 128$	16K	0.005	300	N	90,000	7.5hour

as a standard fluid simulation, up to 10 times slower. In summary, compared to keyframed animation approaches with optimization solution, the computational cost of our fluid rigging and skinning framework is very low, which makes the new method particularly suitable for animating large scenes. Table 2 shows the performance data of all five examples.

5. Conclusion

A general and versatile controlling and editing framework for fluid animation is proposed by extending the concept of rigging and skinning from character animation into fluid animation. The new approach contains two integrated steps: fluid rigging and fluid skinning. The fluid rigging is defined by a point cloud with rigid-body movement and incompressible deformation. The fluid skinning consists of a rig fluidization step that drive the fluid with the fluid rig and a linear flow skinning step that enables adjustments to fine-scale fluid features. Just like rigging and skinning in character animation, fluid rigging allows the animator to achieve intuitive and flexible control, while fluid skinning ensures the controlled fluid to follow the fluid rig and show different fluid features without unnatural artifacts. The proposed fluid rigging-skinning framework is intuitive, efficient, robust and versatile for fluid animation. It support both particle- and grid- solvers and works for both liquid and gas animations.

Our new framework can bring convenience to the editing and controlling of continuous animation, but it is not without limitations. Adjusting the density of point cloud in the fluid rig is important for achieving the desired results: some fluid volumes may be out of control with sparse points, while dense points may cause stiff non-fluid like motions. The current sampling approach relies largely on the experience of the user, and an automated distribution approach would be desirable as a future improvement. In addition, the fluid skinning process is applied after solving the fluid equations and modifies the velocity field, which does risk to introduce an undesirable compression effect. A possible solution is to introduce additional constraints to force incompressibility in the linear flow skinning step, but a naive implementation may bring extra computational cost and it forms an interesting topic for future investigation.

6. Acknowledgement

This work was supported by the National Key Technology R&D Program(Project Number 2017YFB1002701), the Natural Science Foundation of China (Project Number 61521002) and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

[ANSN06] ANGELIDIS A., NEYRET F., SINGH K., NOWROUZEZAHRAI D.: A controllable, fast and stable basis for vortex based smoke simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), Eurographics Association, pp. 25–32. 2

[BHN07] BRIDSON R., HOURIHAM J., NORDENSTAM M.: Curl-noise for procedural fluid flow. In *ACM Transactions on Graphics (ToG)* (2007), vol. 26, ACM, p. 46. 6

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 209–217. URL: http://dl.acm.org/citation.cfm?id=1272690.1272719. 7, 9

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 441–448. doi:10.1145/1015706.1015743. 1, 2

[FM97] FOSTER N., METAXAS D.: Controlling fluid animation. In *Computer Graphics International, 1997. Proceedings* (1997), pp. 178–188. 2

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 15–22. 7, 8, 9

[FSN17] FROST B., STOMAKHIN A., NARITA H.: Moana: Performing water. In *ACM SIGGRAPH 2017 Talks* (New York, NY, USA, 2017), SIGGRAPH '17, ACM, pp. 30:1–30:2. 3

[HK10] HONG J., KIM C.: Controlling fluid animation with geometric potential. *Computer Animation & Virtual Worlds 15*, 3-4 (2010), 147–157. 2

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (Mar. 2014), 426–435. URL: http://dx.doi.org/10.1109/TVCG.2013.105, doi:10.1109/TVCG.2013.105. 7, 8

[IEGT17] INGLIS T., ECKERT M.-L., GREGSON J., THUEREY N.: Primal-dual optimization for fluids. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 354–368. 3

[KTJG08] KIM T., THÜREY N., JAMES D., GROSS M.: Wavelet turbulence for fluid simulation. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 50:1–50:6.

URL: http://doi.acm.org/10.1145/1399504.1360649, doi:10.1145/1399504.1360649. 3

[MBT*15] MERCIER O., BEAUCHEMIN C., THUEREY N., KIM T., NOWROUZEZAHRAI D.: Surface turbulence for particle-based liquid simulations. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 202. 3

[MM13] MADILL J., MOULD D.: Target particle control of smoke simulation. In *Proceedings of Graphics Interface 2013* (Toronto, Ont., Canada, Canada, 2013), GI '13, Canadian Information Processing Society, pp. 125–132. URL: http://dl.acm.org/citation.cfm?id=2532129.2532151. 2

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 449–456. 1, 2

[NB11] NIELSEN M. B., BRIDSON R.: Guide shapes for high resolution naturalistic liquid simulation. *ACM Trans. Graph. 30*, 4 (July 2011), 83:1–83:8. 2, 3

[NC10] NIELSEN M. B., CHRISTENSEN B. B.: Improved variational guiding of smoke animations. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 705–712. 3

[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 *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 217–226. URL: http://doi.acm.org/10.1145/1599470.1599499, doi:10.1145/1599470.1599499. 2, 3

[NSCL08] NARAIN R., SEWALL J., CARLSON M., LIN M. C.: Fast animation of turbulence using energy transport and procedural synthesis. *Acm Transactions on Graphics 27*, 5 (2008), 1–8. 3

[NSG*17] NIELSEN M. B., STAMATELOS K., GRAHAM A., NORDENSTAM M., BRIDSON R.: Localized guided liquid simulations in bifrost. In *ACM SIGGRAPH 2017 Talks* (New York, NY, USA, 2017), SIGGRAPH '17, ACM, pp. 44:1–44:2. doi:10.1145/3084363.3085030. 3

[PCS04] PIGHIN F., COHEN J. M., SHAH M.: Modeling and editing flows using advected radial basis functions. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar Germany, Germany, 2004), SCA '04, Eurographics Association, pp. 223–232. URL: https://doi.org/10.1145/1028523.1028552, doi:10.1145/1028523.1028552. 3

[PHT*13] PAN Z., HUANG J., TONG Y., ZHENG C., BAO H.: Interactive localized liquid motion editing. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 184:1–184:10. 2, 7

[PM17] PAN Z., MANOCHA D.: Efficient solver for spacetime control of smoke. *ACM Trans. Graph. 36*, 5 (July 2017), 162:1–162:13. doi:10.1145/3016963. 1, 3, 8

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

[RLL*13] REN B., LI C. F., LIN M. C., KIM T., HU S. M.: Flow field modulation. *IEEE Trans Vis Comput Graph 19*, 10 (2013), 1708–1719. 3

[RTWT12] RAVEENDRAN K., THUEREY N., WOJTAN C., TURK G.: Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar Germany, Germany, 2012), SCA '12, Eurographics Association, pp. 255–264. URL: http://dl.acm.org/citation.cfm?id=2422356.2422393. 1

[RWTT14] RAVEENDRAN K., WOJTAN C., THUEREY N., TURK G.: Blending liquids. *ACM Trans. Graph. 33*, 4 (July 2014), 137:1–137:10. URL: http://doi.acm.org/10.1145/2601097.2601126, doi:10.1145/2601097.2601126. 3

[SD16] STUYCK T., DUTRÉ P.: Sculpting fluids: A new and intuitive approach to art-directable fluids. In *ACM SIGGRAPH 2016 Posters* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 11:1–11:2. URL: http://doi.acm.org/10.1145/2945078.2945089, doi:10.1145/2945078.2945089. 3

[SS17] STOMAKHIN A., SELLE A.: Fluxed animated boundary method. *ACM Trans. Graph. 36*, 4 (July 2017), 68:1–68:8. 3

[SY05] SHI L., YU Y.: Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), SCA '05, pp. 229–236. 1, 2

[TKPR09] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. *Graph. Models 71*, 6 (Nov. 2009), 221–228. 2, 5

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. *ACM Trans. Graph. 22*, 3 (July 2003), 716–723. doi:10.1145/882262.882337. 1, 2

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. In *ACM Transactions on Graphics (TOG)* (2006), vol. 25, ACM, pp. 1118–1125. 4

[WBZ*17] WANG X.-K., BAN X.-J., ZHANG Y.-L., LIU S.-N., YE P.-F.: Surface tension model based on implicit incompressible smoothed particle hydrodynamics for fluid simulation. *Journal of Computer Science and Technology 32*, 6 (2017), 1186–1197. 7

[WH04] WIEBE M., HOUSTON B.: The tar monster: Creating a character with fluid simulation. In *ACM SIGGRAPH 2004 Sketches* (2004), ACM, p. 64. 3

[WR03] WRENNINGE M., ROBLE D.: Fluid simulation interaction techniques. doi:10.1145/965400.965558. 3

[YCZ11] YUAN Z., CHEN F., ZHAO Y.: Pattern-guided smoke animation with lagrangian coherent structure. In *ACM transactions on graphics (TOG)* (2011), vol. 30, ACM, p. 136. 2

[ZYWL15] ZHANG S., YANG X., WU Z., LIU H.: Position-based fluid control. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (2015), ACM, pp. 61–68. 2