Computational Visual Media
https://doi.org/10.1007/s41095-017-0096-2

Research Article

Editing smoke animation using a deforming grid

Zherong Pan[1] (✉), **Dinesh Manocha**[1]

Abstract We present a new method for editing smoke animations by directly deforming the grid used for simulation. We present a modification to the widely used semi-Lagrangian advection operator and use it to transfer the deformation from the grid to the smoke body. Our modified operator bends the smoke particle streamlines according to the deformation gradient. We demonstrate that the controlled smoke animation preserves the fine-grained vortical velocity components and incompressibility constraints, while conforming to the deformed grid. Moreover, our approach enables interactive 3D smoke animation editing by using a reduced-dimensional subspace. Overall, our method makes it possible to use current mesh editing tools to control the smoke body.

Keywords smoke animation; animation editing; mesh editing

1 Introduction

Realistic smoke effects can greatly enhance visual quality in games and movies. Extensive research into physics-based simulation has resulted in techniques that can efficiently generate realistic smoke animations on current desktop systems. Furthermore, some approaches can perform realtime animations at low or moderate resolutions. However, it is still challenging for animation designers to edit or modify these animations in an intuitive manner to generate plausible results. One such example is illustrated in Fig. 1, where we wish to change the smoke jets so that they rise faster from a chimney, or to bend them to the right. In this paper, we address the problem of smoke animation editing for artistic design.

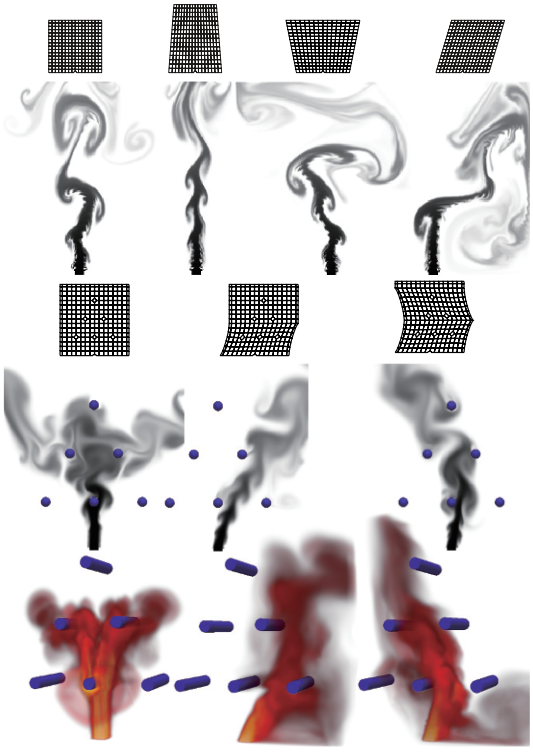

Fig. 1 Our method allows intuitive editing of smoke animations by deforming the simulation grid to bend or stretch 2D smoke jets (top). More complex behavior can be produced by two-pass editing (bottom). The grid resolution is 256×256(×64); we can perform these computations at interactive rates on a PC with an i7-4790 CPU.

Fluid bodies can be discretely represented and numerically modeled either using a set of particles [1] or using a general mesh or grid [2, 3]. In particular, grid-based methods are widely used in software such as Maya, 3DS Max, and Houdini. Using grid representations, several methods have been proposed for modifying and directing smoke animations [4–6]. These methods control smoke bodies using ghost forces. In practice, these ghost forces are difficult

1 The University of North Carolina at Chapel Hill, NC, 27514, USA. E-mail: Z. Pan, zherong@cs.unc.edu (✉); D. Monocha, dm@cs.unc.edu.

Manuscript received: 2017-07-11; accepted: 2017-09-02

to formulate because the smoke animations can be very sensitive to the ghost force templates. As illustrated in Fig. 2, we show that it is difficult to fulfill simple smoke editing requirements using ghost force templates like those in Refs. [4, 7]. Moreover, these methods assume that a target shape exists that the smoke should resemble. However, such a target shape is not readily available in many editing scenarios (see Fig. 1), especially when users want to change the coarse-grained flow patterns while preserving the fine-grained details.

Main results: We present a new method to modify the fluid's velocity field by deforming the underlying grid, as shown in Fig. 3. This underlying grid is not visible in the final animation, but is merely a tool for animation editing. Specifically, we present a modification to the Navier–Stokes equation, so that the deformation is mapped intuitively to the smoke body. This modification is done to the semi-Lagrangian advection operator that bends the streamline using the deformation gradient at that point. This modification can be implemented by simply changing the backtracing direction of the semi-Lagrangian advection operator. In order to preserve the fine-grained vortical velocity components around highly deformed regions, we instead implement this modification as an implicit force term. Finally, incompressibility is preserved using an additional divergence-free projection. We highlight the effectiveness of this transfer operator using a set of editing examples. We can guide smoke flow across boundaries, change flow patterns, and explore variants of smoke animations. The generated animations are plausible, and the fine-grained details are similar to those of uncontrolled animations. Our system can edit 2D animations at interactive framerates on a PC with an i7-4790 CPU. We also present an interactive approach to edit 3D smoke animations by restricting the configuration space of the smoke body to a reduced-dimensional linear-subspace.

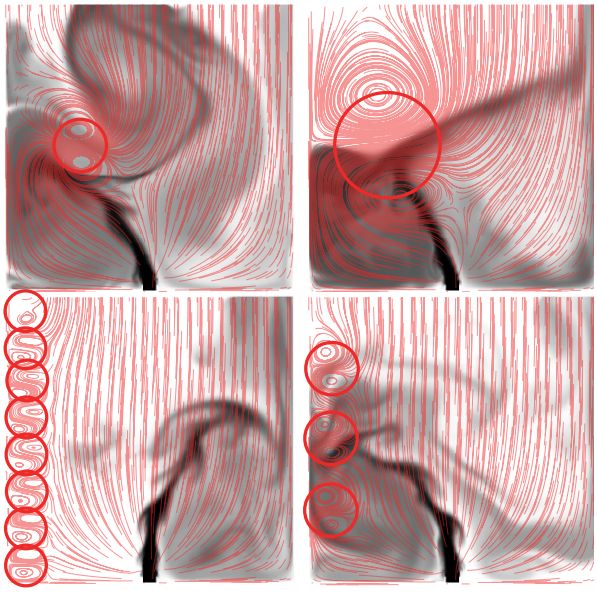

Fig. 2 Difficulty of bending smoke to the right using ghost force templates. Top left: If the template has small support (red circle), smoke suffers from a sharp local push that is visually implausible. Top right: If the template has large support, smoke will bend to the left due to solenoidal constraints (see streamlines in orange). Bottom left: To successfully bend the smoke jet, we need a small array of force templates. Bottom right: If too few ghost force templates are used, the smoke jet can bend in the wrong direction.

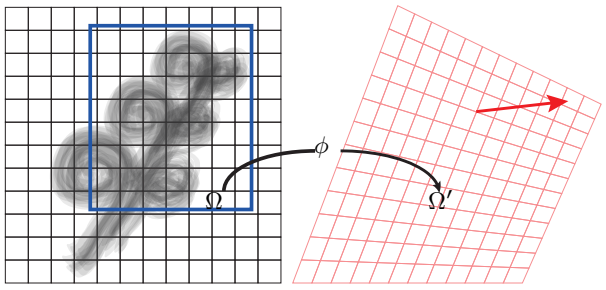

Fig. 3 In our technique, the user edits the smoke animation by deforming the grid from Ω (black) to Ω' (orange) with a single input (red). The effect of the deformation function ϕ is transferred back to the underlying black grid by modifying the fluid advection operator so that the smoke jets rise smoothly to the right (blue).

Our approach has many advantages. Mesh editing is widely studied and good tools are available. We can use those tools directly for smoke editing. Moreover, many mesh editing tools, such as Refs. [8, 9], provide effective ways to regularize the deformed mesh so that sparse and incomplete mesh control inputs may lead to smooth and plausible results. For example, users can drag the mesh at a single vertex and the deformed positions of all other vertices are determined automatically, e.g., by use of as-rigid-as-possible shape manipulation [8] that penalizes any non-rigid deformations.

The rest of the paper is organized as follows. We review previous work in Section 2 and give an overview of fluid dynamics and mesh editing in Section 3. We present the deformation transfer operator in Section 4. In Section 5, we present techniques to accelerate the editing procedure to provide interactive performance. Finally in Section

6, we highlight the performance on complex benchmarks.

2 Related work

Our work builds on a combination of work in mesh editing and smoke animation. We briefly survey prior work in these areas.

Techniques for physics-based animation of fluid, especially smoke and fire, are well-studied. These techniques either discretize fluid-related physical variables on a set of particles [1] or on a mesh [3, 10, 11]. In particular, grid-based methods such as Ref. [3] have been used in content creation software packages. In addition to generating physically plausible animations, we also need techniques to control the smoke. There is considerable work on controlling fluid animations [4–6, 12–18]. These works control the fluid body using a set of spatially-complete keyframe shapes. Sometimes, animators also specify a sparse or incomplete set of control inputs. One example is highlighted in Ref. [7], where the user can drag the liquid surface using a few handles or sketches. However, all of these methods impose controls as ghost force terms, and the generated animations can be sensitive to the formulations of these terms. Instead, we control the fluid body by deforming the simulation grid. Although a deforming grid has been used to handle boundary conditions [19], it has not been used as a fluid control or editing method. Other methods for automatic fluid content creation include Refs. [20, 21] which generate new fluid animations by blending existing animation data, Refs. [22, 23] which guide high-resolution fluid simulations using low-resoultion ones, and Ref. [24] which stylizes fluid animations by adjusting the magnitudes of different modes in the flow field.

Mesh editing is a well-studied problem in computer graphics. Various methods have been developed for mesh editing using different control inputs [9], mesh representations [25], and regularizations [8, 26, 27]; see Ref. [28] for a survey. With these techniques, users can control the mesh using a sparse set of handles and can deform the mesh; uncontrolled degrees-of-freedom are smoothly interpolated in a detail preserving manner. Although our method can be used with any mesh editing tool, we choose the one proposed by Ref. [29] as it can accelerate mesh editing by restricting the configuration space to a reduced-dimensional subspace, and thereby allow interactive editing.

3 Overview

In this section, we first review some fluid simulation methods and then formulate our smoke animation editing problem. Throughout this paper, we consider single-phase fluid bodies such as smoke and fire.

3.1 Single-phase fluid simulation

Our single-phase fluid body is governed by the following incompressible Navier–Stokes equation:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla \boldsymbol{p} + \boldsymbol{f},\ \nabla\cdot\boldsymbol{u} = 0 \\ \dfrac{\partial \rho}{\partial t} + (\boldsymbol{u}\cdot\nabla)\rho = 0 \end{cases} \tag{1}$$

where $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{f}, \rho$ are the velocity field, pressure field, external force field, and passive advected density field, respectively. We assume that the smoke has a homogeneous unit density, which is omitted from the equation. For numerical simulation, the variables in Eq. (1) can be discretized in a Lagrangian manner as a set of particles [1], or in an Eulerian manner as a mesh or grid [3, 10, 11]. Our method assumes that smoke is discretized in an Eulerian manner and we use an approach based on a simple uniform grid [3]. It is possible to use any other grid-based representation instead.

Following Ref. [3], we discretize this equation on a uniform staggered-grid, illustrated by the black grid in Fig. 3. This black grid is our reference domain Ω, on which we store discrete values $\boldsymbol{p}(x_{i,j})$ at each cell center and $\boldsymbol{u}(x_{i-1/2,j}), \boldsymbol{f}(x_{i-1/2,j})$ at each face center, as shown in Fig. 4. Values at other points in Ω are computed by linear interpolation. Using this discretization, Eq. (1) is time-integrated using the fractional step method [2]. Specifically, we accumulate the contributions of the advection term $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$, the pressure term $\nabla\boldsymbol{p}$, and finally the external force term $\boldsymbol{f}$ in separate substeps.

3.2 Smoke animation editing

The goal of our approach is to modify a smoke animation by deforming the domain Ω using a deformation function $\phi(x) : \Omega \to \Omega'$ that maps a point $x \in \Omega$ to a deformed point $\phi(x) \in \Omega'$, as illustrated by the orange grid in Fig. 3. As shown

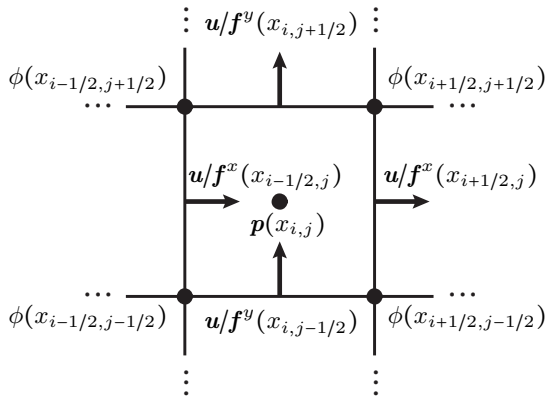

Fig. 4 2D discrete variable arrangements for smoke simulation and mesh editing. We define pressure $\boldsymbol{p}(x_{i,j})$ at each cell center, velocity $\boldsymbol{u}^x(x_{i-1/2,j}), \boldsymbol{u}^y(x_{i,j-1/2})$ and external forces $\boldsymbol{f}^x(x_{i-1/2,j}), \boldsymbol{f}^y(x_{i,j-1/2})$ at each face center, and the deformed positions $\phi(x_{i-1/2,j-1/2})$ at cell corners. Superscripts are coordinate indices.

in Fig. 1, the smoke animation should be modified intuitively and consistently with the deformation. This problem is quite different from one considered in previous works such as Refs. [4, 12], where the current smoke density field, ρ, is pulled towards a target density field, ρ^*. Instead, the goal of our approach is to directly change the coarse-grained components of the velocity field $\boldsymbol{u}$, rather than to perfrom density field matching.

The deformation function $\phi(x)$ is designed using mesh editing tools applied to the black grid shown in Fig. 3. We rely on an efficient mesh editing algorithm [29] to help the user formulate a deformation function $\phi(x)$; any other tools for mesh editing could also be used. We store discrete deformed positions at cell corners $\phi_{i,j} \triangleq \phi(x_{i-1/2,j-1/2})$; the values in each cell are computed using linear interpolation (see also Fig. 4). Ref. [29] computes these $\phi_{i,j}$ as the minimum of the following energy function:

$$\begin{aligned} &\underset{\phi_{i,j}}{\text{argmin}} \quad O(\phi_{i,j}) + \sum_{i,j} C(\phi_{i,j}) \\ &O(\phi_{i,j}) \triangleq \int_\Omega \mu \|\boldsymbol{E}(x)\|_{\text{F}}^2 + \frac{\lambda}{2}\text{tr}(\boldsymbol{E}(x))^2 \text{d}x \end{aligned} \tag{2}$$

where $\boldsymbol{E}(x) \triangleq (\boldsymbol{F}(x)^{\text{T}}\boldsymbol{F}(x) - \boldsymbol{I})/2$ is the Green's strain and $\boldsymbol{F}(x) \triangleq \partial\phi(x)/\partial x$ is the deformation gradient at $x \in \Omega$. Two parameters μ and λ are available allowing the user to change the stiffness of the mesh. The elastic energy $O(\phi_{i,j})$ is discretized using a standard method, such as a finite element method (FEM) with a linear shape function. Finally, $C(\phi_{i,j})$ is the control objective. In this paper, we only consider the point drag control defined as

$$C(\phi_{i,j}) = \frac{I_{i,j}}{2}\|\phi_{i,j} - T_{i,j}\|^2$$

where $T_{i,j}$ is the target position of the control handle on node (i,j) and $I_{i,j}$ indicates whether there is a handle on the given node. After the deformation $\phi(x)$ is computed by solving Eq. (2), the key idea of our method is to transfer this deformation back to the smoke animation.

4 Deformation transfer operator

The goal of our deformation transfer operator is to deform the smoke body so that it mimics the deformation embodied in the function $\phi(x) : \Omega \to \Omega'$. A naive way to implement such an operator is by deforming the density field ρ according to $\phi(x)$ during rendering. However, this transfer operator can be inconvenient for animators because it modifies the simulation domain, including the boundary conditions and obstacles in the physical world, whose shapes are fixed. Therefore, we want a transfer operator that modifies the velocity field $\boldsymbol{u}$ of the smoke body while fixing the simulation domain shape Ω.

Unlike previous methods [12] that change $\boldsymbol{u}$ by modifying the external force $\boldsymbol{f}$, we modify the advection operator $(\boldsymbol{u}\cdot\nabla)$. Compared to modifying the external force, modifying the advection operator is a more intuitive way of editing the smoke body, because such a modification changes the shapes of the streamlines of smoke particles. Our definition of the modified advection operator is based on the observation that if a particle located at $x \in \Omega$ moves with speed $v(x)$, then it should move with speed $\boldsymbol{F}(x)v(x)$ in Ω'. Therefore, we subsitute this new velocity directly into the advection operator to get $(\boldsymbol{F}\boldsymbol{u}\cdot\nabla)$. This modification is similar to the gradient-domain shape deformation [30] where the reconstructed shape is smooth even if the deformation function $\phi(x)$ itself is not. This is illustrated in Fig. 5.

One issue with this formulation of advection operator $(\boldsymbol{F}\boldsymbol{u}\cdot\nabla)$ is that it erroneously takes rigid rotation into account. As illustrated in Fig. 6, since our streamline modification happens at every grid point, an originally straight streamline will be bent into a circular arc under a global rigid rotation of background grid, leading to undesired excessive

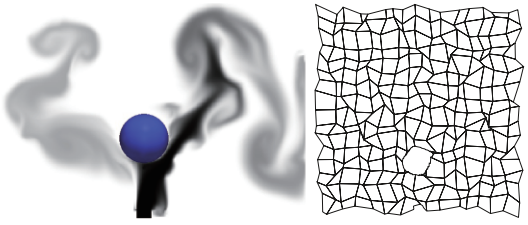

Fig. 5 We randomly perturb the grid (right) to give noisy deformation gradients; this noise is turned into plausible vortical fluid motions.

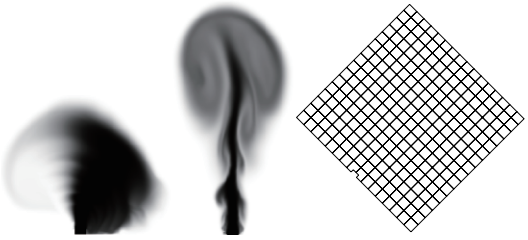

Fig. 6 With just a global rotation (right), the rising smoke jet (center) will be abruptly deformed (left).

deformations. To ensure that only deformation functions $\phi(x)$ that result in non-zero elastic energy $O(\phi_{i,j})$ can deform the smoke body, we follow Ref. [31] and factor out the rigid rotations from $\boldsymbol{F}$ using polar-factorization. Our final formulation of the modified Navier–Stokes equation can be expressed as

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{S}\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla p + \boldsymbol{f} \tag{3}$$

where $\boldsymbol{S}(x) = \boldsymbol{V}\Sigma\boldsymbol{V}^{\mathrm{T}}$ with the singular value decomposition of $\boldsymbol{F}(x)$ being $\boldsymbol{U}\Sigma\boldsymbol{V}^{\mathrm{T}}$.

This modified advection operator can be implemented by precomputing $\boldsymbol{S}$ at each face-center and pre-multiplying $\boldsymbol{u}$ by $\boldsymbol{S}$ before backtracing using a semi-Lagrangian advection operator. However, it is well known that semi-Lagrangian operators introduce excessive numerical dissipation. We can reduce this artifact by implementing the modification as an additional external force term so that Eq. (1) becomes:

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla p + \boldsymbol{f} + \boldsymbol{f}' \\ \boldsymbol{f}' \triangleq ((\boldsymbol{I}-\boldsymbol{S})\boldsymbol{u}\cdot\nabla)\boldsymbol{u} \end{cases} \tag{4}$$

We time-integrate this additional control force term $\boldsymbol{f}'$ implicitly using fixed-point iteration to ensure the stability of the above quadratic form, as outlined in Appendix A. Here we include an adaptive timestep size control scheme to ensure convergence of this iteration. As illustrated in Fig. 7, our experiments show that our approach preserves more vortical velocity components, compared to

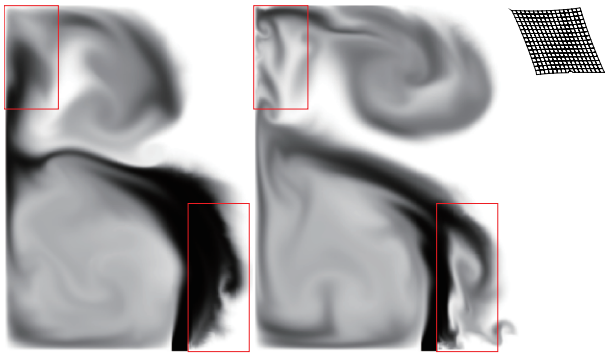

Fig. 7 Modified advection operator implemented as an implicit ghost force term preserves more fine-grained details (right), compared to one implemented as a modification to the semi-Lagrangian advection operator (left). The deformed grid is shown at top right.

using simple modification of the semi-Lagrangian operator. Another method to reduce numerical dissipation is to use a high-order advection operator, following MacCormack. However, our method is more convenient when working with the subspace acceleration techniques introduced in Section 5.

The deformations of the fluid body may introduce compressible velocity components into the system. Currently we simply use an additional divergent-free projection to ensure that $\nabla\cdot\boldsymbol{u} = 0$ everywhere.

5 Subspace acceleration

Using the deformation transfer operator introduced in Section 4, we combine the smoke simulator with a mesh editing tool to provide a smoke animation editing system. Our system allows the user to manipulate the simulation grid and visualize the changed smoke animation at an interactive rate. In practice, obtaining interactive performance can be challenging because we have to solve Eq. (2) to get the deformed grid while time-integrating Eq. (1) at every frame. These operations have complexity polynomial in the number of grid cells. In our experiments, this is possible for editing 2D smoke animations using a high-resolution discretization (up to a grid size of 256×256) but the complexity can be too high for editing 3D smoke animation with a grid size of $256\times256\times64$. In order to further accelerate the editing procedure for 3D animations, we restrict the configuration spaces of the smoke body and the deformable grid to a subspace where all the operations have a complexity polynomial only in the dimension of this subspace.

Specifically, for the grid, we follow Ref. [29]

and assume that its configuration space has a decomposition $\phi = \boldsymbol{U}_m\bar{\phi}$, where ϕ is the discrete deformation function consisting of $\phi_{i,j}$, $\boldsymbol{U}_m^{|\phi|\times p}$ is a set of solenoidal orthogonal bases spanning a p-dimensional linear subspace, and $\bar{\phi}$ gives the corresponding p-dimensional reduced coordinates. Next, we use Galerkin projection and multiply Eq. (2) by $\boldsymbol{U}_m^{\mathrm{T}}$ on the left to give the reduced objective function:

$$\boldsymbol{U}_m^{\mathrm{T}}O(\boldsymbol{U}_m\bar{\phi}) + \boldsymbol{U}_m^{\mathrm{T}}C(\boldsymbol{U}_m\bar{\phi}) \tag{5}$$

After applying the precomputation step proposed in Ref. [29], this objective function can be evaluated with complexity $\boldsymbol{O}(p^4)$ which is much more efficient than evaluating Eq. (2) when $p \ll |\phi|$.

Similarly, we assume that the configuration of the smoke body, the velocity field $\boldsymbol{u}$, has the decomposition $\boldsymbol{u} = \boldsymbol{U}_f\bar{\boldsymbol{u}}$, where $\boldsymbol{u}$ is the discrete velocity field consisting of all face-centered components $\boldsymbol{u}^x(x_{i-1/2,j})$, $\boldsymbol{u}^y(x_{i,j-1/2})$, and $\boldsymbol{U}_f^{|\boldsymbol{u}|\times q}$ is a set of orthogonal bases spanning a q-dimensional linear subspace. Moreover, we assume that all these bases are solenoidal, i.e., $\nabla\cdot\boldsymbol{U}_f = 0$. Finally, $\bar{\boldsymbol{u}}$ gives the q-dimensional reduced coordinates. Similarly, we have the reduced representation $\boldsymbol{f} = \boldsymbol{U}_f\bar{\boldsymbol{f}}$ for the external force as well. By substituting these two representations into Eq. (4) and multiplying by $\boldsymbol{U}_f^{\mathrm{T}}$ on the left, we get the reduced modified Navier–Stokes equation:

$$\frac{\partial\bar{\boldsymbol{u}}}{\partial t} + \mathbf{Adv}(\bar{\boldsymbol{u}}) = \bar{\boldsymbol{f}} \tag{6}$$

We use the orthogonality of $\boldsymbol{U}_f$, and also its solenoidal property to remove $\nabla\boldsymbol{p}$. Finally, we have:

$$\mathbf{Adv}(\bar{\boldsymbol{u}}) = \sum_{i,j,k}\bar{\boldsymbol{u}}_i\bar{\boldsymbol{u}}_j\bar{\phi}_k((\boldsymbol{F}\boldsymbol{U}_{mk})\boldsymbol{U}_{f_i}\cdot\nabla)\mathbf{U}_{fj} \tag{7}$$

where $\boldsymbol{U}_{fi}, \boldsymbol{U}_{mi}$ are the ith columns of bases $\boldsymbol{U}_f$ and $\boldsymbol{U}_m$, respectively, and $\boldsymbol{F}\boldsymbol{U}_{mk}$ consists of the deformation gradients when $\phi = \boldsymbol{U}_{mk}$. This operator is discretized using the midpoint rule introduced in Ref. [32], which preserves details well. The coefficients after $\bar{\boldsymbol{u}}_i\bar{\boldsymbol{u}}_j\bar{\phi}_k$ are constants that can be precomputed; the evaluation of Eq. (6) has complexity $\boldsymbol{O}(q^2p)$ and is very efficient when $p, q \ll |\boldsymbol{u}|$.

Note that the reduced definition $\mathbf{Adv}(\bar{\boldsymbol{u}})$ differs from the original definition of the deformation transfer operator in Section 4, where we used the symmetric part $\boldsymbol{S}$ in place of $\boldsymbol{F}$. It is challenging to use $\boldsymbol{S}$ with subspaces because the polar-factorization cannot be precomputed. To alleviate this problem, we fix a set of vertices on the subspace deformable grid so that global rigid rotation is not possible. We then use the original Eq. (3) to generate the final high-resolution result in the post-processing stage with the edited ϕ.

We summarize our subspace smoke animation editing pipeline in Fig. 8. During the offline stage, we first precompute $\boldsymbol{U}_m$ using modal derivatives as proposed in Ref. [29] and also precompute $\boldsymbol{U}_f$ using principal component analysis (PCA). The data for PCA is collected from simulations under random external forces. During the online editing stage, we time-integrate Eq. (7) and if user editing events are performed, we update $\bar{\phi}$ by minimizing Eq. (5).

6 Results and analysis

In this section, we highlight the editing capability of our method using several benchmarks. All experiments were performed on a desktop PC with an i7-4790 8-core CPU 3.6 GHz and 12 GB of memory. The computational cost of each stage in each benchmark is summarized in Table 1. Without using subspaces, the cost per frame is dominated by solving the linear system for the solenoidal constraints on the smoke body and for solving Eq. (2) using Newton's method. We use a multigrid solver for both linear systems whose cost is $\boldsymbol{O}(n)$ where n is the number of grid cells. For 2D examples, we can achieve interactive response rates. For 3D animations, interactivity is achieved using the subspace techniques introduced

Table 1 Computational cost of each stage of our fluid animation editing system. Left to right: name of benchmark, grid resolution, number of grid bases p, number of fluid bases q, precomputation time, cost of time-integrating Eq. (7) per frame, and cost of solving Eq. (5) per frame. N/A for p or q means that we use fullspace

Benchmark	#Grid	p	q	Precompute	Update Eq. (7)	Update Eq. (5)
2D Smoke Jet (Fig. 1)	256^2	N/A	N/A	0	0.3(s)	0.9(s)
3D Cylinder (Fig. 1)	$256^2\times 64$	20	100	5.2(h)	1.2(s)	0.2(s)
3D Smoke Jet (Fig. 9)	128^3	20	100	4.1(h)	1.5(s)	0.2(s)
3D Smoke Jet (Fig. 9)	128^3	20	20	3.8(h)	1.2(s)	0.2(s)
2D Smoke Circle (Fig. 10)	256^2	N/A	N/A	0	0.4(s)	1.3(s)
2D Maze (Fig. 11)	256×512	N/A	N/A	0	0.9(s)	2.2(s)
3D Maze (Fig. 11)	$128\times 256\times 64$	20	100	4.7(h)	1.3(s)	0.2(s)
2D Eigenmode (Fig. 12)	256^2	N/A	N/A	0	0.3(s)	1.7(s)
3D Eigenmode (Fig. 12)	128^3	N/A	N/A	0	12(s)	71(s)

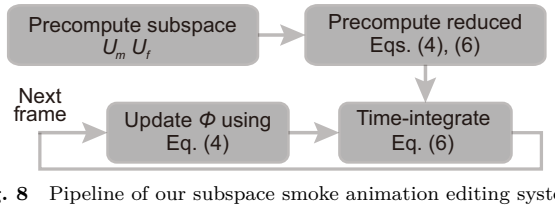

Fig. 8 Pipeline of our subspace smoke animation editing system.

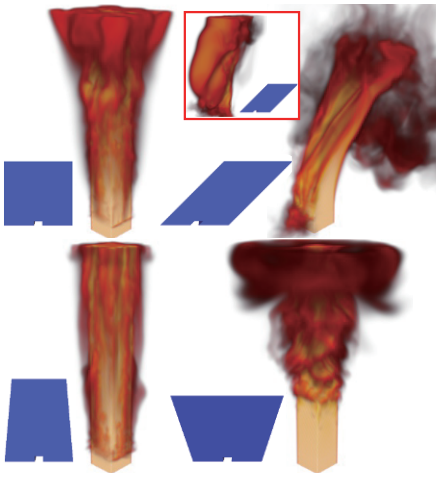

Fig. 9 As in Fig. 1, we show 3D examples of editing smoke jets to bend them (top right), make them rise faster (bottom left), or make the vortices spread wider (bottom right). For all examples, we visualize the edited smoke using subspace simulation with 100 bases. Experiments with a reduced number of bases (e.g., 20, see top sub-image) show that smoke jets do not bend properly with insufficient bases.

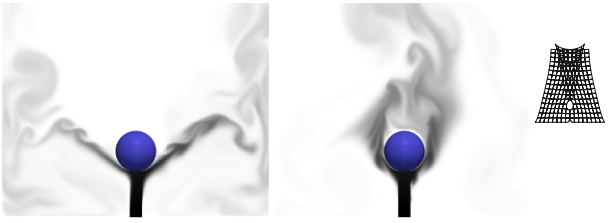

Fig. 10 We deform the grid to merge two branches of the smoke flow after it hits the sphere. Left: the original smoke animation without editing. Middle: the deformed smoke flow after editing. Right: the deformed grid, with discontinuous deformations.

in Section 5. The cost in this case is dominated by $\boldsymbol{O}(p^4 + q^2 p)$ where we use $p, q \leqslant 100$. In all our examples, we use a rectangular uniform grid where each cell can be occupied by either the smoke body or solid boundaries. Cells occupied by solid boundaries or intentionally marked by users are excluded from deformation by setting $\mu = \lambda = 0$ in Eq. (2). Although this causes collapsed or flipped cells during deformation, it does not affect the final fluid animation as we use a conventional semi-Lagrangian operator there.

In our first set of examples, we edit the rising direction and pattern of smoke jets, as illustrated in Figs. 1 and 9. We also compare the 3D editing results achieved using different numbers of bases. We found that 100 bases are sufficient in our benchmarks. Using more bases is usually desirable for an animation editing system to allow more degrees-of-freedom, with a small reduction in performance.

In our second example, we put a solid sphere in the way of rising smoke and deform the grid to guide the smoke flow after hitting the sphere. As illustrated in Fig. 10, the original smoke flow without editing diverges to both sides, but we can merge the two branches by simply deforming the two symmetric parts of the grid. To perform such a large deformation, we cut the grid down the middle to allow discontinuous deformation.

Our third example involves a set of boundaries that cut the simulation domain into four sub-regions. We use our method to guide the smoke to rise to the right and then to the left. In this example, the user performs two passes of editing, which can be done very intuitively as illustrated in Fig. 11.

Our method can not only edit smoke by guiding and bending it, but also automatically generate a set of smoke animations that are different in visually salient ways, as illustrated in Fig. 12. We perform this computation by deforming the grids along each basis in $\boldsymbol{U}_m$. If we compute $\boldsymbol{U}_m$ as the eigenfunctions of Eq. (2) corresponding to small eigenvalues (see Ref. [33]), every perturbation generates plausible but drastically different variants of the original smoke animation.

7 Conclusions and limitations

We have presented a new method to edit smoke animations by deforming the grid used for simulation. This is very convenient for users who are familiar with mesh editing and is more intuitive

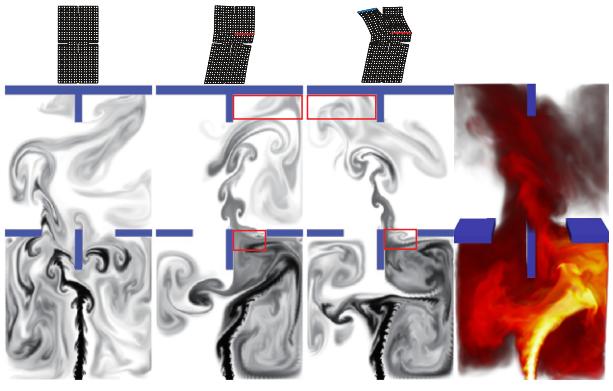

Fig. 11 Smoke jets rise through complex boundaries. Left to right: original smoke animation; we bend the lower part of the grid using red handles, so that smoke only rises through the right opening (lower red block); next we bend the upper part of the grid using blue handles, so that the smoke finally rises through the left opening (top red block); the same approach can be used for 3D animation.

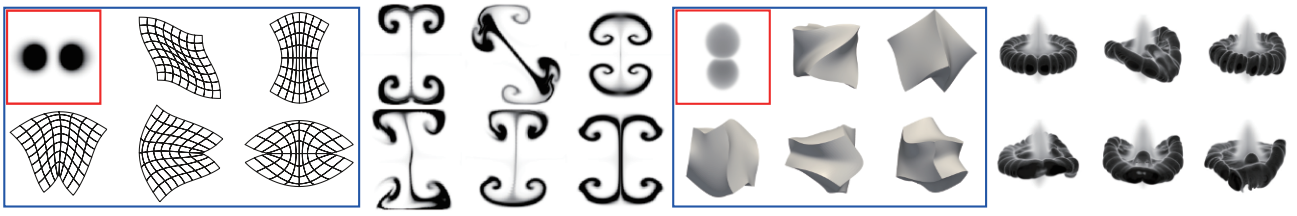

Fig. 12 A set of animations generated by deforming the grid using eigenfunctions of Eq. (2) corresponding to small eigenvalues. Left to right: 2D initial configuration (red) and deformed grids (blue), 2D animation snapshots, 3D initial configuration (red) and deformed grids (blue), and 3D animation snapshots.

than editing a ghost force. In particular, the effective regularization methods for mesh editing are very appealing because such regularizations are not well developed if the editing is performed using ghost forces. Therefore, our method of editing smoke animations using mesh editing tools is more intuitive and can be used with any grid-based method.

The approach transfers the deformation from the grid to the smoke body using a modified advection operator. We have only considered a single-phase fluid, such as smoke or fire. Although the method can also be adopted to interfacial flow such as liquids, we expect that special treatments are needed near the free surface. Such issues can be pursued in the future. For editing 3D animations interactively, we use subspace acceleration techniques.

However, the use of subspace acceleration has limitations. First of all, the editing results can be affected by the choice of the subspace. Appropriate subspace sizes are usually related to specific editing tasks, making such a choice difficult. Moreover, the subspace related precomputation is costly, especially for large and complex scenarios. For example, we cut the grid in the example in Fig. 10. Such cutting is not possible for interactive 3D editing because we need to recompute the subspace bases $\boldsymbol{U}_f$ and $\boldsymbol{U}_m$ after every topological change.

Finally, the editing effects produced by our method have some limitations as well. In general, our method is suitable for making large scale modifications to the flow field, but it cannot be used to edit fine-grained details. In addition, our method cannot change the directions of the smoke velocity field $\boldsymbol{u}$ directly. Instead, we change the velocity fields by scaling the backtracing velocity $\boldsymbol{u}$ along different directions specified by the symmetric matrix $\boldsymbol{S}$. The symmetry of $\boldsymbol{S}$ will scale both $\boldsymbol{u}$ and $-\boldsymbol{u}$ with the same ratio; this may be inconvenient when users want unilateral scaling.

A promising direction of future work is to consider using other user interfaces for editing fluid animations. Besides mesh deformation, there are many other user interfaces in the community, such as sketch-based interfaces [34]. The main challenge in using these new interfaces is to convert user inputs such as sketches into fluid control inputs, e.g., forces or keyframes, in an intuitive manner.

Appendix A Algorithm

We summarize the algorithm for applying the implicit ghost force in Algorithm A1.

Acknowledgements

This research is supported in part by Army Research Office and National Science Foundation.

References

[1] Müller, M.; Charypar, D.; Gross, M. Particle-based fluid simulation for interactive applications. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 154–159, 2003.

[2] Stam, J. Stable fluids. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, 121–128, 1999.

[3] Fedkiw, R.; Stam, J.; Jensen, H. W. Visual simulation of smoke. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 15–22, 2001.

[4] Treuille, A.; McNamara, A.; Popović, Z.; Stam, J. Keyframe control of smoke simulations. *ACM Transactions on Graphics* Vol. 22, No. 3, 716–723, 2003.

[5] Raveendran, K.; Wojtan, C.; Turk, G. Hybrid smoothed particle hydrodynamics. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 33–42, 2011.

[6] Shi, L.; Yu, Y. Taming liquids for rapidly changing targets. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 229–236, 2005.

Algorithm A1 We implicitly apply the ghost force term $\boldsymbol{f}'$ using a fixed point iteration given in Line 10. This iteration sometimes does not converge using a large timestep, so we use an adaptive timestepping scheme controlled by the number of substeps n.

Input: a velocity field $\boldsymbol{u}$ and the timestep size Δt
Output: the velocity field $\boldsymbol{u}^*$ after applying external force $\boldsymbol{f}'$

1: ▷ initial guess
2: $\boldsymbol{u}^* := \boldsymbol{u}$
3: ▷ initialize number of substeps
4: $n := 1$
5: **while** true **do**
6: ▷ decompose time-integration into n substeps
7: **for** $t := 1, \cdots, n$ **do**
8: **for** $k := 1, \cdots, 10$ **do**
9: ▷ solve for $\boldsymbol{u}^{**}$ using fixed point iterations
10: set $\boldsymbol{u}^{**} := \boldsymbol{u} + ((\boldsymbol{I} - \boldsymbol{S})\boldsymbol{u}^* \cdot \nabla)\boldsymbol{u}^* \Delta t/n$
11: ▷ see if we have converged
12: **if** $\|\boldsymbol{u}^{**} - \boldsymbol{u}^*\| < 10^{-6}$ **then**
13: break
14: **else**
15: $\boldsymbol{u}^* := \boldsymbol{u}^{**}$
16: **end if**
17: **end for**
18: ▷ increase number of timesteps if not convergent
19: **if** $k := 10$ **then**
20: $n := 2n$
21: break
22: **end if**
23: **end for**
24: ▷ time-integration accomplished
25: **if** $t = n$ **then**
26: break
27: **end if**
28: **end while**

[7] Pan, Z.; Huang, J.; Tong, Y.; Zheng, C.; Bao, H. Interactive localized liquid motion editing. *ACM Transactions on Graphics* Vol. 32, No. 6, Article No. 184, 2013.

[8] Igarashi, T.; Moscovich, T.; Hughes, J. F. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics* Vol. 24, No. 3, 1134–1141, 2005.

[9] Jacobson, A.; Baran, I.; Popović, J.; Sorkine-Hornung, O. Bounded biharmonic weights for real-time deformation. *Communications of the ACM* Vol. 57, No. 4, 99–106, 2011.

[10] Klingner, B. M.; Feldman, B. E.; Chentanez, N.; O'Brien, J. F. Fluid animation with dynamic meshes. *ACM Transactions on Graphics* Vol. 25, No. 3, 820–825, 2006.

[11] Losasso, F.; Gibou, F.; Fedkiw, R. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics* Vol. 23, No. 3, 457–462, 2004.

[12] Fattal, R.; Lischinski, D. Target-driven smoke animation. *ACM Transactions on Graphics* Vol. 23, No. 3, 441–448, 2004.

[13] McNamara, A.; Treuille, A.; Popović, Z.; Stam, J. Fluid control using the adjoint method. *ACM Transactions on Graphics* Vol. 23, No. 3, 449–456, 2004.

[14] Thürey, N.; Keiser, R.; Pauly, M.; Rüde, U. Detail-preserving fluid control. *Graphical Models* Vol. 71, No. 6, 221–228, 2009.

[15] Sato, S.; Dobashi, Y.; Yue, Y.; Iwasaki, K.; Nishita, T. Incompressibility-preserving deformation for fluid flows using vector potentials. *The Visual Computer* Vol. 31, Nos. 6–8, 959–965, 2015.

[16] Sato, S.; Dobashi, Y.; Iwasaki, K.; Ochiai, H.; Yamamoto, T.; Nishita, T. An optimization approach for designing fluid flow fields. In: Proceedings of the 31st Computer Graphics International Short Paper, 2014.

[17] Raveendran, K.; Thuerey, N.; Wojtan, C.; Turk, G. Controlling liquids using meshes. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 255–264, 2012.

[18] Pan, Z.; Manocha, D. Efficient solver for spacetime control of smoke. *ACM Transactions on Graphics* Vol. 36, No. 5, Article No. 162, 2017.

[19] Feldman, B. E.; O'Brien, J. F.; Klingner, B. M.; Goktekin, T. G. Fluids in deforming meshes. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 255–259, 2005.

[20] Raveendran, K.; Wojtan, C.; Thuerey, N.; Turk, G. Blending liquids. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 137, 2014.

[21] Thuerey, N. Interpolations of smoke and liquid simulations. *ACM Transactions on Graphics* Vol. 36, No. 1, Article No. 3, 2016.

[22] Nielsen, M. B.; Bridson, R. Guide shapes for high resolution naturalistic liquid simulation. *ACM Transactions on Graphics* Vol. 30, No. 4, Article No. 83, 2011.

[23] Nielsen, M. B.; Christensen, B. B. Improved variational guiding of smoke animations. *Computer Graphics Forum* Vol. 29, No. 2, 705–712, 2010.

[24] Ren, B.; Li, C.-F.; Lin, M. C.; Kim, T.; Hu, S.-M. Flow field modulation. *IEEE Transactions on Visualization and Computer Graphics* Vol. 19, No. 10, 1708–1719, 2013.

[25] Levi, Z.; Levin, D. Shape deformation via interior RBF. *IEEE Transactions on Visualization and Computer Graphics* Vol. 20, No. 7, 1062–1075, 2014.

[26] Botsch, M.; Pauly, M.; Gross, M. H.; Kobbelt, L. PriMo: Coupled prisms for intuitive surface modeling. In: Proceedings of the Eurographics Symposium on Geometry Processing, 11–20, 2006.

[27] Zhuang, Y.; Canny, J. Haptic interaction with global deformations. In: Proceedings of the IEEE

International Conference on Robotics and Automation, 2428–2433, 2000.

[28] Botsch, M.; Sorkine, O. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics* Vol. 14, No. 1, 213–230, 2008.

[29] Barbič, J.; James, D. L. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics* Vol. 24, No. 3, 982–990, 2005.

[30] Huang, J.; Shi, X.; Liu, X.; Zhou, K.; Wei, L.-Y.; Teng, S.-H.; Bao, H.; Guo, B.; Shum, H.-Y. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics* Vol. 25, No. 3, 1126–1134, 2006.

[31] Irving, G.; Teran, J.; Fedkiw, R. Tetrahedral and hexahedral invertible finite elements. *Graphical Models* Vol. 68, No. 2, 66–89, 2006.

[32] Liu, B.; Mason, G.; Hodgson, J.; Tong, Y.; Desbrun, M. Model-reduced variational fluid simulation. *ACM Transactions on Graphics* Vol. 34, No. 6, Article No. 244, 2015.

[33] Hauser, K. K.; Shen, C.; O'Brien, J. F. Interactive deformation using modal analysis with constraints. In: Proceedings of Graphics Interface, 247–255, 2003.

[34] Olsen, L.; Samavati, F. F.; Sousa, M. C.; Jorge, J. A. Sketch-based modeling: A survey. *Computers & Graphics* Vol. 33, No. 1, 85–103, 2009.

Zherong Pan is a Ph.D. candidate in the Computer Science Department of the University of North Carolina at Chapel Hill. He received his B.Sc. degree from Shanghai Jiao Tong University majoring in software engineering in 2011. His research interests include physics-based animation and visualization.

Dinesh Manocha is currently Phi Delta Theta/Matthew Mason Distinguished Professor of Computer Science at the University of North Carolina at Chapel Hill. He received his B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi, in 1987, and his Ph.D. degree in computer science from the University of California at Berkeley in 1992. He has coauthored more than 420 papers in leading conferences and journals on computer graphics, robotics, and scientific computing. Prof. Manocha has received awards including an IBM Fellowship, Alfred P. Sloan Fellowship, NSF Career Award, Office of Naval Research Young Investigator Award, SIGMOD IndySort Winner, Honda Research Award, Hettleman Award at UNC Chapel Hill, and 14 best paper awards at leading conferences. He is a Fellow of the ACM, AAAS, and IEEE and has received a Distinguished Alumnus Award from the Indian Institute of Technology, Delhi.