Efficient Solver for Spacetime Control of Smoke

Zherong Pan Dinesh Manocha [1]
Department of Computer Science
the University of North Carolina

We present a novel algorithm to control the physically-based animation of smoke. Given a set of keyframe smoke shapes, we compute a dense sequence of control force fields that can drive the smoke shape to match several keyframes at certain time instances. Our approach formulates this control problem as a PDE-constrained spacetime optimization. In order to compute the locally optimal control forces, we alternatively optimize the velocity fields and density fields using an ADMM optimizer. In order to reduce the high complexity of multiple passes of fluid resimulation during velocity field optimization, we utilize the coherence between consecutive fluid simulation passes. We demonstrate the benefits of our approach by computing accurate solutions on 2D and 3D benchmarks. In practice, we observe up to an order of magnitude improvement over prior optimal control methods.

Additional Key Words and Phrases: Fluid Simulation, Optimal Control

1. INTRODUCTION

Physically-based fluid animations are widely used in computer graphics and related areas. Over the past few years, research in fluid simulation has advanced considerably and it is now possible to generate plausible animations for movies and special effects in a few hours on current desktop systems. In this paper, we mainly deal with the problem of the keyframe-based spacetime control of smoke, a special kind of fluid. Given a set of keyframe smoke shapes, our goal is to compute a dense sequence of control forces such that the smoke can be driven to match these keyframes at certain time instances. This problem is an example of directable animation and arises in different applications, including special effects [Rasmussen et al. 2004] (to model a character made of liquid) or artistic animations [Angelidis et al. 2006] (to change the moving direction of the smoke plume). Some of these control techniques, such as [Nielsen and Bridson 2011], are used in the commercial fluid software.

In practice, the keyframe-based control of fluids is still regarded as a challenging problem. Unlike fluid simulation, which deals with the problem of advancing the current fluid state to the next one by time integrating the Navier-Stokes equations, a fluid controller needs to consider an entire sequence of fluid states that results in a high dimensional space of possible control forces. For example, to control a 3D smoke animation discretized on a uniform grid at resolution 64^3 with 60 timesteps, the dimension of the resulting space of control forces can be as high as 10^8. The problem of computing the appropriate control force sequence in such a high dimensional space can be challenging for any continuous optimization algorithm. Furthermore, the iterative computation of control forces would need many iterations, each of which involves solving a 2D or 3D fluid simulation problem that can take hours on a desktop system.

Fluid control problems have been well studied in computer graphics and animation. At a broad level, prior techniques can be classified into proportional-derivative (PD) controllers and optimal controllers. PD controllers [Fattal and Lischinski 2004; Shi and Yu 2005] guide the fluid body using additional ghost force terms that are designed based on a distance measure between the current fluid shape and the keyframe. On the other hand, optimal controllers [Treuille et al. 2003; McNamara et al. 2004] formulate the problem as a spacetime optimization over the space of possible control forces constrained by the fluid governing equations, i.e., the Navier-Stokes equations. The objective function of this optimization formulation consists of two terms: The first term requires the fluid shape to match the keyframe shape at certain time instances, while the second term requires the control force magnitudes to be as small as possible.

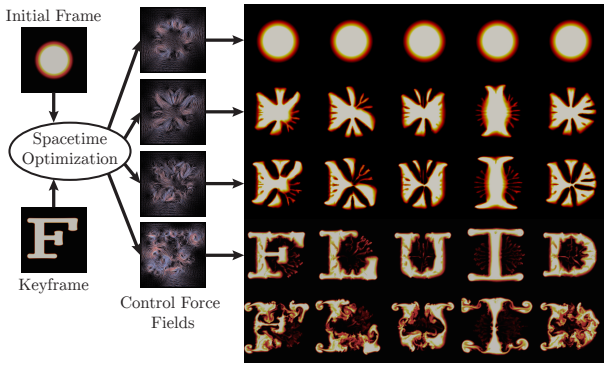

Fig. 1: Given the keyframe, we use spacetime optimization to compute a dense sequence of control force fields, matching a smoke ball to the word "FLUID". We highlight the control force fields. Five such animations are generated, at resolution 128^2 with 40 timesteps. Each of these optimization computations take about half an hour on a desktop PC, and is about 17 times faster than conventional gradient-based optimizer.

Optimal controllers are advantageous over PD controllers in that they are less sensitive to the choice of the parameters and they search for the control forces with the smallest possible magnitude, which usually provides smoother keyframe transitions as well as satisfies the fluid dynamic constraints. Treuille et al. [2003] and McNamara et al. [2004] use a simple gradient-based optimizer to search for control forces. This method is easy to implement, but can be computationally inefficient since each gradient evaluation needs to solve a fluid simulation problem. Such repeated fluid simulations slow down the overall performance. In the original work [Treuille et al. 2003], this issue is alleviated by reducing the dimension of the search space using a set of control force templates. However, this treatment also restricts the amount of fluid-like details in the controlled animations.

We present a new, efficient optimization algorithm for controlling smoke. Our approach exploits the special structure of the Navier-Stokes equations discretized on a regular staggered grid, and solves the optimization problem by finding the stationary point of the first order optimality (Karush-Kuhn-Tucker) conditions [Nocedal and Wright 2006]. However, unlike prior methods [Treuille et al. 2003; McNamara et al. 2004] that only solve for the primal variables, we maintain both the primal and dual variables (i.e., the

Lagrangian multipliers). By maintaining the additional dual variables, we can iteratively update our solution without requiring it to satisfy the Navier-Stokes equations exactly in each iteration. Therefore, we alternatively update the velocity fields and control force fields using the alternating direction method of multiplier (ADMM) [Boyd et al. 2011]. In order to update the control force fields efficiently without fluid resimulation, we present a nonlinear multigrid solver: the full approximation scheme (FAS) [Brandt and Livne 2011]. Using a novel spacetime smoothing operator that takes all the timesteps into consideration, our multigrid can converge within a number of iterations independent of the grid resolution and the number of timesteps.

We have evaluated our approach on several benchmarks. Our benchmarks vary in terms of the grid resolution, the number of timesteps, and the control force regularization parameter. We highlight results with up to 60 timesteps at the resolution of 64^3. Without using force templates, we allow each component of the velocity field to be controlled. In practice, our algorithm can compute a convergent animation in less than 50 iterations, and the overall runtime performance is about an order of magnitude faster than a gradient-based quasi-Newton optimizer [Nocedal and Wright 2006] for similar accuracy. An example of achieved smooth transitions between keyframes is illustrated in Figure 1.

2. RELATED WORK

In this section, we give a brief overview of prior techniques for fluid simulation, multigrid solvers and animation control algorithms.

Fluid simulation has been an active area of research in both computer graphics and computational fluid dynamics. The simulation of fluid is typically solved by a discretized time integration of the Navier-Stokes equations or their equivalent forms. At a broad level, prior fluid simulators can be classified into Lagrangian or Eulerian solvers according to the discretization of the convection operator. In order to model smoke and fire, a purely Eulerian solver [Fedkiw et al. 2001] is the standard technique. In terms of free-surface flow, hybrid Lagrangian-Eulerian representation [Zhu and Bridson 2005] has been widely used in computer graphics. In our work, we confine ourselves to the control of fluids without free-surface, i.e., smoke or fire. We use [Harlow and Welch 1965] as our underlying fluid simulator.

Multigrid solvers are widely used for fluid simulation. Multigrid is a long-standing concept that has been widely used to efficiently solve linear systems discretized from elliptic partial differential equations (see [Brandt and Livne 2011]). This idea has been successfully applied to fluid simulation [Chentanez et al. 2007; Chentanez and Müller 2011; Zhang and Bridson 2014] to find the solenoidal component of the velocity field. In terms of PDE-constrained optimization and control theory, the idea of multigrid acceleration has been extended to the spatial temporal domain. Borzi and Griesse [2005] proposed a semi-coarsening spacetime multigrid to control the time-dependent reaction-diffusion equation. Hinze et al. [2012] used a spacetime multigrid to solve the velocity tracking problem governed by the Navier-Stokes equations. The nonlinear multigrid used in our method is closely related to [Hinze et al. 2012], which also solves a spacetime optimization problem. However, our method is a spatial-only multigrid with the smoothing operator that handles all the timesteps at once. Moreover, unlike [Hinze et al. 2012], which solves a velocity tracking problem, our formulation is a density tracking problem. Therefore, the multigrid is used as a subproblem solver in our ADMM optimization framework.

Fluid control problems tend to be challenging and computationally demanding. Compared to other kinds of animations, e.g., character locomotion [Mordatch et al. 2012], the configuration space of fluid body is of much higher dimension. Prior work in this area can be classified into two categories: PD controllers [Fattal and Lischinski 2004; Shi and Yu 2005] and optimal controllers [Treuille et al. 2003; McNamara et al. 2004]. PD controllers compute the control forces by considering only the configuration of the fluid at the current and next time instance. For example, in [Shi and Yu 2005], a PD controller is used where the control forces are made proportional to the error between the current fluid shape and the target keyframe shape. Similar ideas are used for controlling smoke [Fattal and Lischinski 2004] and liquid [Shi and Yu 2005; Raveendran et al. 2012]. In contrast, optimal controllers search for a sequence of control forces that minimize an objective function. Prior methods [Treuille et al. 2003; McNamara et al. 2004] typically solve spacetime optimization over a high-DOF search space to compute such control forces. Recently, these two methods have been combined [Pan et al. 2013] by first optimizing for the fluid shape at each keyframe and then propagating the changes to the neighboring timesteps. Fluid control can also be achieved by combining or modifying the results of existing fluid simulation data [Raveendran et al. 2014] or guiding fluid using a designed or captured low-resolution animation [Nielsen and Bridson 2011; Nielsen and Christensen 2010; Gregson et al. 2014].

Our algorithm is based on spacetime optimization, similar to [Treuille et al. 2003; McNamara et al. 2004]. In order to solve this optimization problem, we use the ADMM method [Boyd et al. 2011]. This solver has also been previously used in [Gregson et al. 2014] for fluid capturing and guiding. However, our method differs from these previous works in three ways. First, unlike [McNamara et al. 2004], whose method solves for primal variables only, we use a primal-dual formulation. This treatment does not require the Navier-Stokes equations to be satisfied exactly in each iteration of optimization. Moreover, to solve fluid tracking problem, [Gregson et al. 2014] uses ADMM method to address incompressible constraints in one timestep. While we use ADMM method for spacetime optimization taking all timesteps into consideration. Finally, [Gregson et al. 2014] considers the linear solenoidal constraints as hard constraints. In our work, we take into account the entire nonlinear fluid governing equations as the hard constraints, resulting in a nonlinear subproblem that is solved using a multigrid method.

3. FLUID CONTROL

In this section, we formulate the spacetime fluid control problem based on fluid dynamics (Section 3.1) and optimal control theory (Section 3.2). The set of symbols used throughout the paper can be found in Figure 3, and the subscript i is the timestep index. In general, we are dealing with a dynamic system whose configuration space is denoted as s_i at physical time $i\Delta t$. Consecutive configurations s_i and s_{i+1} are related by the partial differential equation denoted as the function f: $s_{i+1} = f(s_i, u_i, \Delta t)$, where u_i is the control input. An optimal controller computes a set of control inputs $\{u_i | i = 0, \cdots, N-1\}$ that minimize the objective function denoted as function $E(s_0, \cdots, s_N)$. The overall optimal control problem is specified using the pair of functions f and E. In the case of smoke control problems, f is a discretization of the Navier-Stokes equations, and E measures the difference between the smoke and keyframe shapes at certain time instances.

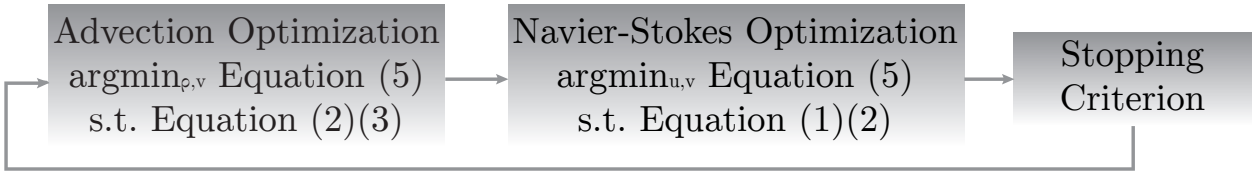

Fig. 2: Our Algorithm Pipeline: We use the ADMM method to decompose the problem into two subproblems: Advection Optimization (AO) which only considers passive advection as hard constraints; and Navier-Stokes Optimization (NSO) which only considers fluid dynamics as hard constraints.

Symbol	Meaning
v_i	velocity field
u_i	ghost force field
ρ_i	density or dye field
p_i	pressure field
s_i	state vector
$\mathbf{Adv}$	self advection operator
$\mathbf{A}$	passive advection operator
Δt	timestep size
N	number of timesteps

(a) Symbols for fluid dynamic system

Symbol	Meaning
v_i^*	slack velocity field
λ_i	augmented Lagrangian multiplier field
μ_i	Lagrangian multiplier for passive advection
γ_i	Lagrangian multiplier for $\nabla \cdot v_i^* = 0$
$\bar{p}_i$	Lagrangian multiplier for $\nabla \cdot v_i = 0$
K	penalty coefficient for constraint $v_i = v_i^*$
r	regularization coefficient for u_i
c_i	indicator of keyframe at timestep i
C_i	metric measure for density field
$\mathbf{R}$	restriction operator of FAS
$\mathbf{P}$	prolongation operator of FAS
$\mathbf{S}$	smoothing operator of FAS
$\mathbf{Q}$	solenoidal projection operator

(b) Symbols for spacetime optimization

Fig. 3: Symbol table.

3.1 Fluid Dynamic System

In our work, we restrict ourselves to the control of incompressible fluids without a free surface. Fluids such as smoke and fire, which are commonly used in movies and animations, fall into this category. We omit viscous terms for brevity. Small viscosity can be handled by a slight modification to f, which does not increase the complexity of our algorithm. Following [Harlow and Welch 1965; Pavlov et al. 2011], we discretize the velocity-vorticity version of the Navier-Stokes equations using finite difference scheme and backward Euler time integrator for advection. Our configuration space $s_i = \left({v_i}^T \, {p_i}^T \, {\rho_i}^T\right)^T$ concatenates the velocity field v_i, the kinetic pressure field p_i, and the density or dye field ρ_i. These scalar and vector fields are discretized on a staggered grid, which has been widely used by previous works such as [Fedkiw et al. 2001]. The transfer function f under such discretization can be represented as:

$$\frac{v_{i+1} - v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] = u_i - \nabla p_{i+1}, \tag{1}$$

$$\nabla \cdot v_{i+1} = 0, \tag{2}$$

$$\rho_{i+1} = \mathbf{A}\left[\rho_i, v_i\right], \tag{3}$$

where the self-advection operator $\mathbf{Adv}\left[\bullet\right]$ is a discretization of the quadratic operator $\nabla \times \bullet \times \bullet$ and we assume constant unit fluid density. The pressure field p_{i+1} is identified with the Lagrangian multiplier of the divergence free constraints $\nabla \cdot v_{i+1} = 0$. Finally, the operator $\mathbf{A}\left[\bullet, \bullet\right]$ is the passive scalar advection operator discretized as: $\rho_{i+1} = \mathbf{e}^{\mathbf{A}(v_i)\Delta t}\rho_i$, where matrix $\mathbf{A}(v_i)$ is the second order upwinding stencil [Leonard 1979]. By approximating the matrix exponential using Taylor series, the advection operator can be defined as:

$$\mathbf{A}\left[\rho_i, v_i\right] = \sum_{k=0}^{\infty} \frac{\Delta t^k}{k!} \mathbf{A}(v_i)^k \rho_i. \tag{4}$$

When k tends to infinity, this upwinding advection operator is unconditionally stable since $\mathbf{A}(v_i)$ is skew-symmetric, so that $\mathbf{e}^{\mathbf{A}(v_i)\Delta t}$ is an orthogonal matrix and $\|\rho_{i+1}\| = \|\rho_i\|$. In practice, we truncate k to a finite value. Specifically, we set k adaptively to be the smallest integer satisfying $\frac{\Delta t^k}{k!}\mathbf{A}(v_i)^k \rho_i < 1e^{-5}$. Although this operator is computationally more expensive than the widely used semi-Lagrangian operator, it generates smoother controlled animations with large timestep size, as shown in Figure 4. This is useful when fewer timesteps are used to reduce the runtime cost.

3.2 Spacetime Optimization

The optimal control of the dynamic system, discussed in Section 3.1, can be formulated as a spacetime optimization over the configuration trajectory $s_0, \cdots, s_N$. Our objective function is similar to the ones proposed in prior works [Treuille et al. 2003; McNamara et al. 2004] that try to match ρ_i to a set of keyframes ρ_i^* while minimizing the magnitude of control forces u_i. The overall optimization problem can be formulated as:

$$\begin{aligned} \underset{u_i}{\mathbf{argmin}} \quad & E = \frac{1}{2}\sum_{i=0}^{N} c_i \|\rho_i - \rho_i^*\|^2 + \frac{r}{2}\sum_{i=0}^{N-1} \|u_i\|^2 \\ \mathbf{s.t.} \quad & s_{i+1} = f(s_i, u_i, \Delta t), \end{aligned} \tag{5}$$

where c_i is 1 if there is a keyframe ρ_i^* at frame i and 0 otherwise. r is the regularization coefficient of the control forces.

Treuille et al. [2003] and McNamara et al. [2004] solve this optimization by eliminating the transfer function f and plugging them into the objective function. Although this reformulation simplifies the problem into an unconstrained optimization, their new objective function takes a much more complex form, which is a long chain of function compositions. To minimize the new objective function, Treuille et al. [2003] and McNamara et al. [2004] use a general-purpose gradient-based optimizer. A typical gradient-based optimizer such as the Quasi-Newton method [Byrd et al. 1995] requires repeated gradient calculation to approximate the Hessian matrix and performs line search to compute the stepsize. Each such gradient calculation requires a fluid resimulation, which becomes the major bottleneck in their algorithm.

3.3 Our Approach

Prior methods require that the solution computed during each iteration should satisfy the Navier-Stokes equations exactly, i.e., is a feasible solution. As a result, each iteration takes considerable running time. In practice, this requirement can be overly conservative because we only need to ensure that the final computed solution at the end of the algorithm is feasible. Thus, we can relax this requirement during the intermediate steps, and only need to ensure that the final solution lies in the feasible domain. This is a well-known idea and has been used by many other numerical optimization algorithms such as the interior point method [Nocedal and Wright 2006].

Based on this observation, we design a new optimization algorithm illustrated in Figure 2. We first notice that our objective function is essentially constrained by two kinds of partial differential equations: the passive advection (Equation 3) governing the time evolution of the density field ρ_i; and the incompressible Navier-Stokes (Equation 1 and Equation 2) governing the time evolution of the velocity field v_i. We introduce a set of slack variables to break these two kinds of constraints into two subproblems: Advection Optimization (AO) is constrained only by Equation 3 and Navier-Stokes Optimization (NSO) is constrained only by Equation 1 and Equation 2. In order to solve the Advection Optimization (Section 4.1), we use a fixed point iteration defined for its KKT conditions. For the Navier-Stokes Optimization (Section 4.2), we update our solution using the full approximation scheme (FAS) to avoid repeated fluid resimulations. This leads to considerable speedup over prior methods, not only because of the fast convergence of our multigrid solver, but also because the multigrid solver allows warm-starting so that we can utilize coherence between consecutive iterations. In contrast, previous methods use fluid resimulations, which always solve Navier-Stokes equations from scratch, and solve them exactly.

4. SPACETIME OPTIMIZATION

In this section, we present our novel algorithm to solve Equation 5. By introducing a series of slack variables v_i^*, we can decompose the overall optimization problem into two subproblems and reformulate Equation 5 as:

$$\begin{aligned} \underset{u_i}{\mathbf{argmin}} \quad & \frac{1}{2}\sum_{i=0}^{N} c_i \|\rho_i - \rho_i^*\|^2 + \frac{r}{2}\sum_{i=0}^{N-1}\|u_i\|^2 + \\ & \lambda_i^T(v_i - v_i^*) + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2 \qquad (6) \\ \mathbf{s.t.} \quad & \frac{v_{i+1}-v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] = u_i - \nabla p_{i+1} \\ & \rho_{i+1} = \mathbf{A}\left[\rho_i, v_i^*\right], \quad \nabla \cdot v_i = 0, \end{aligned}$$

where we added the augmented Lagrangian term $\lambda_i^T(v_i - v_i^*)$ and the penalty term $\frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2$. This kind of optimization can be solved efficiently using the well-known alternating direction method of multipliers (ADMM) [Boyd et al. 2011]. Specifically, in each iteration of our algorithm, we first fix v_i, p_i and solve for v_i^*. This subproblem is denoted as the Advection Optimization (AO) because the PDE constraints are just passive advection of the density field ρ_i. We then fix v_i^* and solve for v_i, p_i. We denote this subproblem as the Navier-Stokes Optimization (NSO), constrained by the incompressible Navier-Stokes equations. The final step is to adjust λ_i according to the constraint violation as: $\lambda_i = \lambda_i + K\beta(v_i - v_i^*)$ where β is a constant parameter.

The idea of introducing slack variables to relax hard constraints has been used in several previous works, e.g., for fluid tracking [Gregson et al. 2014] and projective dynamic simulation [Narain et al. 2016]. The advantage of decomposing the problem up is that we can derive simple and effective algorithms to solve each subproblem. Our algorithm directly solves the first order optimality (KKT) conditions of both problems. To solve the AO subproblem, we introduce a fixed point iteration in Section 4.1, while for the NSO subproblem, which is the bottleneck of the algorithm, we introduce the nonlinear multigrid solver (FAS) in Section 4.2.

4.1 Advection Optimization

The goal of solving the AO subproblem is to find a sequence of velocity fields v_i^* to advect ρ_i so that it matches the keyframes, assuming that these v_i^* are uncorrelated. By dropping terms irrelevant to v_i^* from Equation 6, we get a concise formulation for the AO subproblem:

$$\begin{aligned} \underset{v_i^*}{\mathbf{argmin}} \quad & \frac{1}{2}\sum_{i=0}^{N}(\rho_i - \rho_i^*)^T C_i(\rho_i - \rho_i^*) + \qquad (7) \\ & \frac{K}{2}\sum_{i=0}^{N-1}\|v_i + \lambda_i/K - v_i^*\|^2 \\ \mathbf{s.t.} \quad & \rho_{i+1} = \mathbf{A}\left[\rho_i, v_i^*\right] \quad \nabla \cdot v_i^* = 0, \end{aligned}$$

where we can absorb the augmented Lagrangian term $\lambda_i^T(v_i - v_i^*)$ by setting: $v_i \leftarrow v_i + \lambda_i/K$.

Due to the inherent nonlinearity and ambiguity in the advection operator, an AO solver is prone to falling into local minimum, leading to trivial solutions. We introduce two additional modifications to Equation 7 to avoid these trivial solutions. First, we replace the scalar coefficient c_i with a matrix C_i which could be used to avoid the problem of a zero gradient if the keyframe ρ_i^* is far from the given density field ρ_i. Similar to [Treuille et al. 2003; Fattal and Lischinski 2004], we use the idea of Gaussian Pyramid [Adelson et al. 1984] and take $C_i = c_i\Sigma_k G_k^T G_k$ to be a series of Gaussian filters G_k with receding support. Specifically, G_k has a standard deviation of $\sigma(G_k) = 2\sigma(G_{k-1})$. The Gaussian Pyramid makes our method almost resolution invariant, since any local error in the density field will always lead to a non-zero gradient value at every point in the grid domain. We also introduce additional solenoidal constraints on v_i^*. Note that this term does not alter the optima of Equation 6 since $v_i = v_i^*$ on convergence. However, it prevents the optimizer from creating or removing densities in order to match the keyframe, which is a tempting trivial solution.

We solve this optimization via a fixed point iteration derived from its KKT conditions. To derive this system we introduce Lagrangian multipliers μ_i for each advection equation $\rho_{i+1} = \mathbf{A}\left[\rho_i, v_i^*\right]$ and γ_i for the solenoidal constraints, giving a Lagrangian function:

$$\begin{aligned} \mathcal{L} = \frac{1}{2}\sum_{i=0}^{N}(\rho_i - \rho_i^*)^T C_i(\rho_i - \rho_i^*) + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2 + \\ \sum_{i=0}^{N-1}\mu_i^T(\rho_{i+1} - \mathbf{A}\left[\rho_i, v_i^*\right]) + \gamma_i^T \nabla \cdot v_i^*. \end{aligned}$$

After taking the derivative of the above Lagrangian against ρ_i, v_i^* (primal variables) and μ_i, γ_i (dual variables), respectively, we get the following set of KKT conditions for $0 \le i \le N$:

$$\begin{aligned} \mu_{i-1} &= \frac{\partial \mathbf{A}\left[\rho_i, v_i^*\right]}{\partial \rho_i}^T \mu_i - C_i(\rho_i - \rho_i^*) \\ v_{i-1}^* &= \mathbf{Q}(v_{i-1} + \frac{\partial \mathbf{A}\left[\rho_{i-1}, v_{i-1}^*\right]}{\partial v_{i-1}^*}^T \frac{\mu_{i-1}}{K}) \qquad (8) \\ \rho_{i+1} - \mathbf{A}\left[\rho_i, v_i^*\right] &= 0, \quad \nabla \cdot v_i^* = 0, \end{aligned}$$

where we set $\mu_{-1} = \mu_N = 0$ to unify the index, and we have replaced γ_i with a solenoidal projection operator $\mathbf{Q}$. This actually defines a fixed point iteration where we can first update ρ_i in a forward pass and then update μ_i, v_i in a backward pass. This is closely related to the adjoint method [McNamara et al. 2004], which also

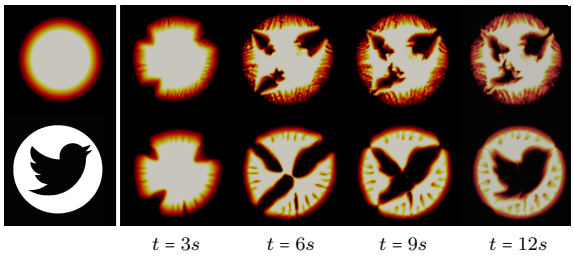

Fig. 4: We tested the fixed point iteration Equation 8 using different advection operator $\mathbf{A}[\bullet,\bullet]$ to deform an initially circle-shaped smoke (top left) into the bird icon (bottom left). The AO subproblem solved using the semi-Lagrangian operator involves lots of popping artifacts (top row). The upwinding operator in Equation 4 does not suffer from such problems (bottom row).

takes a forward-backward form. Unlike [McNamara et al. 2004] which then solves v_i^* using quasi-Newton method, a fixed point iteration is much simpler to implement, and a general-purpose optimizer is not needed. The most costly step in applying Equation 8 is the operator $\mathbf{Q}$ where we use conventional multigrid Poisson solver [Brandt and Livne 2011].

A pseudo-code of our AO solver is given in Algorithm 1. We have introduced two additional strategies to guarantee the convergence of the fixed point iteration. First, we determine the order of Taylor expansion (k in Equation 4) in the forward pass (Line 11) and fix this k value in the backward pass to ensure that the order of expansion is fixed within each gradient estimation. Moreover, we introduce a simple line search strategy (Line 14 to Line 28).

In the above derivation, since we do not exploit any structure in the operator $\mathbf{A}[\rho_i, v_i^*]$, basically any advection operator other than Equation 4, such as semi-Lagrangian, could be used as long as its partial derivatives against ρ_i, v_i^* are available. Empirically, however, Equation 4 generally gives smoother animations especially under large timestep size. This is because the semi-Lagrangian operator can jump across multiple cells when performing backtracking, and the density value changes in these cells are ignored. As a result, the semi-Lagrangian operator suffers from popping artifacts as illustrated in Figure 4, while our operator (Equation 4), being purely grid-based, does not exhibit such problem. Unlike [Treuille et al. 2003], where these popping artifacts can be alleviated by constraining control force fields to a small set of force templates, we allow every velocity component to be controlled. In this case, the use of our new advection operator is recommended.

4.2 Navier-Stokes Optimization

Complementary to Section 4.1, the goal of the Navier-Stokes Optimization is to enforce the correlation between v_i given the sequence of guiding velocity fields v_i^*. The optimization takes the following form:

$$\begin{aligned} \underset{v_i}{\mathbf{argmin}} \quad & \frac{r}{2}\sum_{i=0}^{N-1}\|u_i\|^2 + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2 \\ \mathbf{s.t.} \quad & \frac{v_{i+1}-v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] = u_i - \nabla p_{i+1} \\ & \nabla \cdot v_i = 0. \end{aligned}$$

This subproblem is the bottleneck of our algorithm, for which a forward-backward adjoint method similar to Equation 8 requires solving the Navier-Stokes equations exactly in the forward pass. To avoid this costly solve, we update primal as well as dual variables from the previous iteration using a unified algorithm. In the same way as in Section 4.1, we derive the KKT conditions and assemble them into a set of nonlinear equations:

$$f = \begin{pmatrix} \frac{K}{r}(v_i - v_i^*) + \frac{\partial u_i}{\partial v_i}^T u_i + \frac{\partial u_{i-1}}{\partial v_i}^T u_{i-1} + \nabla \bar{p}_i \\ \nabla \cdot v_i \\ \frac{v_{i+1}-v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] - u_i + \nabla p_{i+1} \\ \nabla \cdot u_i \end{pmatrix} = 0, \forall i \quad (9)$$

where the partial derivatives are $\frac{\partial u_i}{\partial v_i} = -\frac{I}{\Delta t}$, $\frac{\partial u_{i-1}}{\partial v_i} = \frac{I}{\Delta t} + \frac{\partial \mathbf{Adv}[v_i]}{\partial v_i}$, and the additional variable $\bar{p}_i$ is the Lagrangian multiplier for the solenoidal constraint: $\nabla \cdot v_i = 0$. We refer readers to Appendix A for the derivation of Equation 9. In summary, we have to solve for the primal variables u_i, v_i as well as the dual variables $p_i, \bar{p}_i$. Unlike Equation 8, however, we do not differentiate these

Algorithm 1 The Fixed Point Iteration: This is used to solve the AO subproblem. The algorithm consists of a forward sweep that updates the density fields ρ_i and a backward sweep that updates μ_i and v_i^*. Here, we introduce a line-search parameter α to ensure algorithm convergence, where v_i^{**} stores the tentative solution.

Input: Initial v_i, ρ_0, $\alpha \in (0,1]$, and keyframes ρ_i^*
Output: Fixed point solution v_i^*, μ_i

1: $E \leftarrow \infty$
2: **for** $i = 0, \cdots, N-1$ **do**
3: ▷ Initialization
4: $v_i^* \leftarrow v_i$
5: $v_i^{**} \leftarrow v_i$
6: **end for**
7: **while** not converged **do**
8: ▷ Forward pass
9: **for** $i = 1, \cdots, N$ **do**
10: ▷ Find primal variables ρ
11: Find smallest k such that $\frac{\Delta t^k}{k!}\mathbf{A}(v_{i-1}^{**})^k \rho_{i-1} < 1e^{-5}$
12: $\rho_i \leftarrow \mathbf{A}[\rho_{i-1}, v_{i-1}^{**}]$
13: **end for**
14: ▷ Ensure function value decrease
15: $E^{new} \leftarrow \frac{1}{2}\sum_{i=0}^{N}\|\rho_i - \rho_i^*\|_{C_i}^2 + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^{**}\|^2$
16: **if** $E^{new} < E$ **then**
17: $E \leftarrow E^{new}$
18: **for** $i = 1, \cdots, N-1$ **do**
19: $v_i^* \leftarrow v_i^{**}$
20: **end for**
21: increase α
22: **else**
23: **for** $i = 1, \cdots, N-1$ **do**
24: $v_i^{**} \leftarrow v_i^*$
25: **end for**
26: decrease α
27: **goto** *Line* 8
28: **end if**
29: ▷ Backward pass
30: set $\mu_{-1} \leftarrow 0$, $\mu_N \leftarrow 0$
31: **for** $i = N, \cdots, 1$ **do**
32: ▷ Find dual variables μ
33: $\mu_{i-1} \leftarrow \frac{\partial \mathbf{A}[\rho_i, v_i^*])}{\partial \rho_i}^T \mu_i - C_i(\rho_i - \rho_i^*)$
34: ▷ Find primal variables v
35: $v_{i-1}^{**} \leftarrow (1-\alpha)v_{i-1}^* + \alpha\mathbf{Q}(v_{i-1} + \frac{\partial \mathbf{A}[\rho_{i-1}, v_{i-1}^*])}{\partial v_{i-1}^*}^T \frac{\mu_{i-1}}{K})$
36: **end for**
37: **end while**

two sets of variables and solve for them by iteratively bringing the residual f to zero.

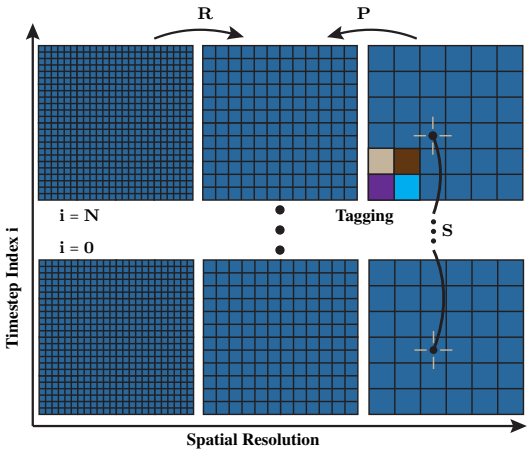

Fig. 5: A 2D illustration of our full approximation scheme (FAS). We use semi-coarsening only in the spatial direction (horizontal), with each finer level doubling the grid resolution. We use trilinear interpolation operators for $\mathbf{P}, \mathbf{R}$ and tridiagonal SCGS smoothing for $\mathbf{S}$, which solves the primal variables v_i, u_i (defined on faces as short white lines) and dual variables $p_i, \bar{p}_i$ (defined in cell centers as black dots) associated with one cell across all the timesteps (vertical) by solving a block tridiagonal system. The solve can be made parallel by the $8-color$ tagging in 3D or $4-color$ tagging in 2D.

To this end, we develop a full approximation scheme (FAS), which is a geometric multigrid algorithm designed for solving a nonlinear system of equations as illustrated in Figure 5. The multigrid solver is a classical tool originally used for solving linear systems induced from elliptical PDEs. Recently, the multigrid solver has also been used for solving nonlinear problems such as PDE-constrained optimization. For example, Hinze et al. [2012] used a spacetime multigrid to track and stabilize the velocity field of incompressible Navier-Stokes flow, and Borzi et al. [2005] used a semi-coarsening multigrid to control a reaction-diffusion flow. The nonlinear multigrid in our formulation can be considered as a combination of these two approaches: a semi-coarsening multigrid to control the incompressible Navier-Stokes flow. Given that we want to solve a density tracking problem, instead of a velocity tracking problem, we use the spatial FAS multigrid as our NSO subproblem solver. We refer the readers to [Brandt and Livne 2011] for a detailed introduction and briefly review the core idea in the following parts.

4.3 Full Approximation Scheme

Since Equation 9 is valid for all the indices i, we concatenate all the timestep-related variables and discard subscripts for convenience. A multigrid solver works on a hierarchy of grids in descending resolutions. In each FAS iteration, it refines the solution $(v, \bar{p}, u, p)$ by reducing the residual $f(v, \bar{p}, u, p)$. Since different components of the residual can be reduced most effectively at different resolutions, the multigrid solver downsamples the residual to the appropriate resolutions and then upsamples and combines their solutions. With properly defined operators introduced in this section, our multigrid algorithm can generally achieve a linear rate of error reduction, which is optimal in the asymptotic sense.

To adopt this idea to solve Equation 9, we introduce a hierarchy of spatial grids $(v^h, \bar{p}^h, u^h, p^h)$, where h is the cell size. We use semi-coarsening in spatial direction only where every coarser level doubles the cell size. We denote the coarser level as $(v^{2h}, \bar{p}^{2h}, u^{2h}, p^{2h})$, and use the simple FAS-VCycle(2,2) iteration to solve the nonlinear system of equations: $f(v, \bar{p}, u, p) = \mathbf{res}$. See Algorithm 2 for details of the NSO solver.

Algorithm 2 FAS $\mathbf{VCycle}(v^h, \bar{p}^h, u^h, p^h, \mathbf{res}^h)$: This is used to solve the NSO subproblem. The algorithm is a standard FAS-VCycle with 2 pre and post smoothing (Line 8, Line 28) and 10 final smoothing (Line 3).

Input: A tentative solution $(v^h, \bar{p}^h, u^h, p^h)$
Output: Refined solution to $f(v^h, \bar{p}^h, u^h, p^h) = \mathbf{res}^h$

1: **if** h is coarsest **then**
2: ▷ Final smoothing for the coarsest level
3: **for** $k = 1, \cdots, 10$ **do**
4: $\mathbf{S}(v^h, \bar{p}^h, u^h, p^h)$
5: **end for**
6: **else**
7: ▷ Pre smoothing
8: **for** $k = 1, 2$ **do**
9: $\mathbf{S}(v^h, \bar{p}^h, u^h, p^h)$
10: **end for**
11: ▷ Down-sampling
12: **for** $t = v, \bar{p}, u, p$ and **do**
13: $t^{2h} \leftarrow \mathbf{R}(t^h)$
14: $t^h \leftarrow t^h - \mathbf{P}(t^{2h})$
15: **end for**
16: ▷ Compute FAS residual by combining:
17: ▷ 1. the solution on coarse resolution
18: ▷ 2. the residual on fine resolution
19: $\mathbf{res}^{2h} \leftarrow f(v^{2h}, \bar{p}^{2h}, u^{2h}, p^{2h})$
20: $\mathbf{res}^{2h} \leftarrow \mathbf{res}^{2h} + \mathbf{R}(\mathbf{res}^h - f(v^h, \bar{p}^h, u^h, p^h))$
21: ▷ VCycle recursion
22: $\mathbf{VCycle}(v^{2h}, \bar{p}^{2h}, u^{2h}, p^{2h}, \mathbf{res}^{2h})$
23: ▷ Up-sampling
24: **for** $t = v, \bar{p}, u, p$ **do**
25: $t^h \leftarrow t^h + \mathbf{P}(t^{2h})$
26: **end for**
27: ▷ Post smoothing
28: **for** $k = 1, 2$ **do**
29: $\mathbf{S}(v^h, \bar{p}^h, u^h, p^h)$
30: **end for**
31: **end if**

The fast convergence of the geometric FAS relies on a proper definition of the three application-dependent operators: $\mathbf{R}, \mathbf{P}$ and $\mathbf{S}$. The restriction operator $\mathbf{R}$ downsamples a fine grid solution to a coarser level for efficient error reduction, and the prolongation operator $\mathbf{P}$ upsamples the coarse grid solution to correct the fine grid solution. We use simple trilinear interpolation for these two operators whether applied on scalar or vector fields. Finally, designing the smoothing operator $\mathbf{S}$ is much more involved. $\mathbf{S}$ should, by itself, be a cheap iterative solver for $f(v, \bar{p}, u, p) = \mathbf{res}$. Compared with previous works such as [Chentanez and Müller 2011] where multigrid is used for solving the pressure field p only, we are faced with two new challenges. First, since we are solving the primal as well as dual variables, which gives a saddle point problem, the Hessian matrix is not positive definite in the spatial domain, so that a

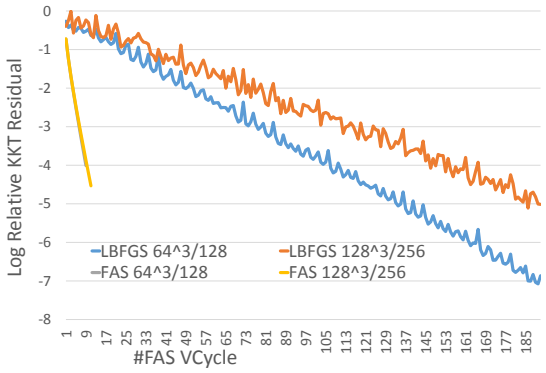

Fig. 6: Convergence history of FAS compared with that of the LBFGS optimizer, running on two grid resolutions and with a different number of timesteps (denoted as n^d/N). FAS achieves a linear rate of error reduction independent of grid resolution and number of timesteps, as the two curves overlap.

Jacobi or Gauss-Seidel (GS) solver does not work. Second, we are not coarsening in the temporal domain, so the temporal correlation must be considered in the smoothing operator.

Our solution is to consider the primal and dual variables at the same time using the Symmetric Coupled Gauss-Seidel (SCGS) smoothing operator [Vanka 1983]. SCGS smoothing is a primal-dual variant of GS. In our case, where all the variables are stored in a staggered grid, SCGS smoothing considers one cell at a time. It solves the primal variables v, u stored on the 6 cell faces as well as the dual variables $p, \bar{p}$ stored in the cell center at the same time by solving a small 14×14 linear problem (10×10 in 2D). Like red-back-GS smoothing, we can parallelize SCGS smoothing using the 8-color tagging (see Figure 5).

The above SCGS solver only considers one timestep at a time. To address the second problem of temporal correlation, we augment the SCGS solver with the temporal domain. We solve the 14 variables associated with a single cell across all the timesteps at once. Although this involves solving a large $14N \times 14N$ linear system for each cell, the left hand side of the linear system is a block tridiagonal matrix so that we can solve the system in $\mathcal{O}(N)$. Indeed, the Jacobian matrix of f takes the following form:

$$\frac{\partial f}{\partial v,\bar{p},u,p} = \begin{pmatrix} \frac{K}{r}I & \nabla & \frac{\partial u_0}{\partial v_0}^T & & \\ \nabla^T & & & & \\ \hline \frac{\partial u_0}{\partial v_0} & & -I & \nabla & \frac{\partial u_0}{\partial v_1} \\ & & \nabla^T & & \\ \hline & & \frac{\partial u_0}{\partial v_1}^T & & \frac{K}{r}I \quad \nabla \\ & & & & \nabla^T \quad \ddots \end{pmatrix}, \tag{10}$$

where the size of each block is 5×5 in 2D and 7×7 in 3D. Due to this linear time solvability, the optimal multigrid performance is still linear in the number of spatial-temporal variables. The average convergence history for our multigrid solver is compared with a conventional LBFGS algorithm [McNamara et al. 2004] in Figure 6. Our algorithm achieves a stable linear rate of error reduction independent of both the grid resolution and the number of timesteps.

4.4 ADMM Outer Loop

Equipped with solvers for the two subproblems, we present our ADMM outer loop in Algorithm 3. We find it very time-consuming for either Equation 8 or a quasi-Newton method solving the AO subproblem to converge to an arbitrarily small residual due to the non-smooth nature of the operator $\mathbf{A}[\bullet, \bullet]$. Both algorithms decrease the objective function in the first few iterations and then wander around the optimal solution. In view of this, we run Equation 8 (Section 4.1) for a fixed number of iterations before moving on to the NSO subproblem (Section 4.2) so that each ADMM iteration has $\mathcal{O}(n^d N)$ complexity and is linear in the number of space-time variables. Finally, our stopping criterion for the NSO subproblem is that the residual $\|f\|_\infty < \epsilon_{FAS}$. Our stopping criterion for the ADMM outer loop is that the maximal visual difference, the largest difference of the density field over all the timesteps, generated by two consecutive ADMM iterations should be smaller than ϵ_{ADMM}.

Algorithm 3 ADMM Outer Loop

Input: Parameters $K, r, \rho_i^*, \epsilon_{STFAS}, \epsilon_{ADMM}$
Output: Optimized velocity fields v_i and density fields ρ_i

for $i = 0, \cdots, N$ **do**
  Set $v_i \leftarrow 0$
  Set $\rho_i^{last} \leftarrow \rho_i$
end for
while true **do**
  ▷ Solve the AO subproblem
  Run Algorithm 1 for a fixed number of iterations
  ▷ Solve the NSO subproblem
  while $\|f(v, \bar{p}, u, p)\|_\infty > \epsilon_{STFAS}$ **do**
    Algorithm 2
  end while
  ▷ Stopping criterion
  if $\max_i \|\rho_i^{last} - \rho_i\|_\infty < \epsilon_{ADMM}$ **then**
    Return v_i, ρ_i
  end if
  for $i = 0, \cdots, N$ **do**
    Set $\rho_i^{last} \leftarrow \rho_i$
    ▷ Update augmented Lagrangian multiplier
    Set $\lambda_i \leftarrow \lambda_i + K\beta(v_i - v_i^*)$
  end for
end while

5. RESULTS AND ANALYSIS

Name	Value
Δt	$0.4 \sim 2.0s$
K	10^3
r	$10^{2\sim4}$
β for updating λ_i	1
#Equation 8	2
ϵ_{FAS}	10^{-5}
ϵ_{ADMM}	$\frac{\rho_{max}}{100}$

Table I. : Parameters.

Parameter Choice: We use the same set of parameters listed in Table I for all experiments, where ρ_{max} is the maximal density magnitude at the initial frame. In our experiments, the ADMM algorithm always converges in fewer than 50 iterations. Further, running only 2 iterations of Equation 8 in each ADMM loop will not deteriorate the performance. In

fact, according to the averaged convergence history of the AO subproblem illustrated in Figure 7, the fixed point iteration Equation 8 usually converges in the first 4 iterations before it wanders around a local minimum. After fine tuning, we found that 2 iterations lead to the best overall performance. In this case, the overhead of solving the AO subproblem is marginal compared with the overhead of solving the NSO subproblem. Finally, unlike fluid simulation, the performance of spacetime optimization does not depend on the timestep size due to our robust advection operator (Equation 4). When we increase the timestep size from $0.4s$ to $2s$ for the examples in Figure 8 and Figure 9, which is extremely large, our algorithm's convergence behavior is about the same. Under this setting, the convergence history of the ADMM outer loop for our first example (Figure 1) is illustrated in Figure 7. The convergence history can be decomposed into two stages. In the first stage, the first term of Equation 5 (keyframe shape matching) dominates, the solver gradually evolves the solution to match the keyframe shape, and the KKT-Residual is not monotonically decreasing. In the second stage, however, the second term (control force regularization) dominates Equation 5 and the KKT-Residual quickly decreases. Since the solutions of consecutive ADMM iterations do not change much, we have also tried to use just a few SCGS smoothing steps, instead of the entire FAS Algorithm 2, to approximately solve the NSO subproblem. In practice, we observed this treatment smoothed out the fluid-like behaviors, when large regularization r is used.

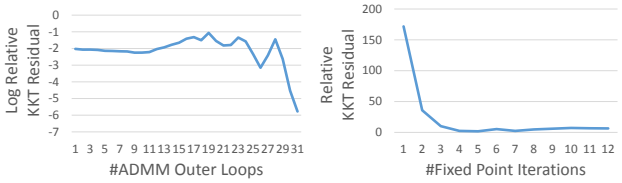

Fig. 7: We profile the convergence history of the example Figure 1. We plot the logarithm of relative KKT residual of the optimized velocity field after each ADMM loop (left); and the absolute residual of the AO subproblem's KKT conditions after each iteration of Equation 8 (right).

Benchmarks: To demonstrate the efficiency and robustness of our algorithm, we used 7 benchmark problems that vary in their grid resolution, number of timesteps, and number of keyframes. The memory overhead and computational overhead are summarized in Table II. All of the results are generated on a desktop PC with an i7-4790 8-core CPU 3.6GHz and 12GB of memory. We use OpenMP for multithread parallelization.

Our first example is five controlled animations matching a circle to the letters "FLUID". Compared with [Treuille et al. 2003], which uses a relatively small set of control force templates to reduce the search space of control forces, we allow control on every velocity component so that the matching to keyframe is almost exact. After the keyframe, we remove the control force, and rich smoke details are generated by pure simulation as illustrated in Figure 1. However, in the controlled phase of Figure 1, this example seems "too much controlled", meaning that most smoke-like behaviors are lost. This effect has also been noticed in [Treuille et al. 2003]. However, unlike their method, in which the number of templates needs to be carefully tuned to recover such behavior, we can simply adjust the regularization r in our system to balance matching exactness and the amount of smoke-like behaviors. In Figure 8, we generated three animations with two keyframes: first two circles and then a bunny, using $r = 10^{2,3,4}$ respectively. These animations are also shown in the video. Our algorithm is robust to a wide range of parameter choices. But more iterations are needed for the multigrid to converge for a larger r as shown in Table II. Finally, since we allow every velocity component to be optimized, the resulting animation exhibits lots of small-scale details as indicated in Figure 9, which is not possible with the small set of force templates used in [Treuille et al. 2003].

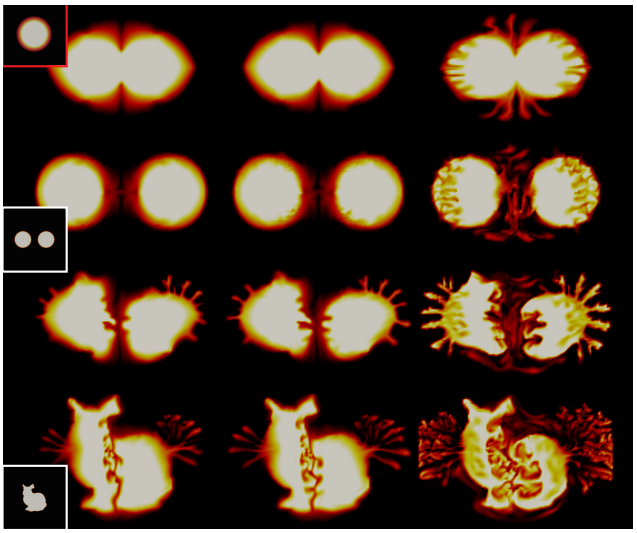

Fig. 8: For this animation, we match the circle (red) first to two smaller circles and then to a bunny (we show frames $20, 40, 60, 80$ from top to bottom). The resolution is $128^2/80$, and we test three different values of ghost force regularization $r = 10^{2,3,4}$ (from left to right). More smoke-like behaviors are generated as we increase r.

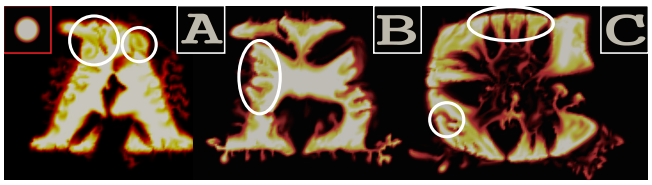

Fig. 9: In this example, we deform a sphere into letter "A", then letter "B" and finally letter "C". For such complex deformation, it is advantageous to allow every velocity component to be optimized. So that a lot of fine-scale details can be generated as illustrated in the white circles.

In addition to these 2D examples, we also tested our algorithm on some 3D benchmarks. Our first example is shown in Figure 10 and runs at a resolution of $64^3/40$. We use two keyframes at frame 20 and 40, and the overall optimization takes about 7 hours. In our second example, shown in Figure 11, we try to track the smoke with a dense sequence of keyframes from the motion capture data of a human performing a punch action. Such an example is considered the most widely used benchmarks for PD-type controllers such as [Shi and Yu 2005]. With such strong and dense guidance, our algorithm converges very quickly, within 5 iterations. Our third example (Figure 13) highlights the effect of regularization coefficient r in 3D. Like our 2D counterpart Figure 8, larger r usually results in more wake flow behind moving smoke bodies. Finally, we evaluated our algorithm on a benchmark with keyframe shapes of varying genera. As illustrated in Figure 12, the initial smoke shape has genus zero, but we use two keyframes, where the smoke shapes

Example(n^d/N)	Boundary	#ADMM	Avg. AO (s)	Avg. NSO (s)	Total(hr)	Memory(Gb)	Total LBFGS(hr)
Letters "FLUID"($128^2/40, r = 10^3$)	Neumann	13	10	60	0.25	0.06	4
Letters "FLUID"($128^2/80, r = 10^3$)	Neumann	17	21	142	0.76	0.12	9
Circle Bunny($128^2/80, r = 10^2$)	Neumann	25	20	130	1.04	0.2	12
Circle Bunny($128^2/80, r = 10^3$)	Neumann	37	20	220	2.46	0.2	15
Circle Bunny($128^2/80, r = 10^4$)	Neumann	43	20	218	2.84	0.2	16
Letters ABC($128^2/60, r = 10^4$)	Neumann	33	16	179	1.78	0.15	14
Sphere Armadillo Bunny($64^3/40, r = 10^3$)	Neumann	17	103	1341	6.81	1.34	N/A
Varying Genus($64^2 \times 32/40, r = 10^3$)	Periodic	20	82	840	5.12	0.67	N/A
Human Mocap($64^2 \times 128/60, r = 10^3$)	Periodic	5	1437	3534	6.9	4.0	N/A
Moving Sphere($64^3/60, r = 10^2$)	Neumann	17	630	1792	11.43	2.2	N/A
Moving Sphere($64^3/60, r = 10^3$)	Neumann	22	630	1978	15.93	2.2	N/A
Letters ABC 3D($64^3/150, r = 10^2$)	Periodic	20	1512	3220	26.28	5.9	N/A

Table II. : Memory and computational overhead for all the benchmarks. From left to right: name of example (resolution parameters); the spatial boundary condition; number of outer ADMM iterations; average time spent on each AO subproblem; average time spent on each NSO subproblem; total time until convergence using our algorithm; memory overhead; total time until convergence using LBFGS. By comparing the three "Circle Bunny" examples, we can observe that the number of ADMM outer loops is roughly linear to $log_{10}(r)$. More ADMM outer loops are needed, if more fluid-like behaviors are desired. From the two examples of the Letters "FLUID" (Line 1 and Line 2), we can observe that the computational cost of each ADMM outer loop (Avg. AO + Avg. NSO) is roughly linear in the number of timesteps. This cost is also governed by the number of keyframes. By comparing Line 2 and Line 4, we can observe that the Circle Bunny example which involves two keyframes requires more computation to solve the NSO subproblem.

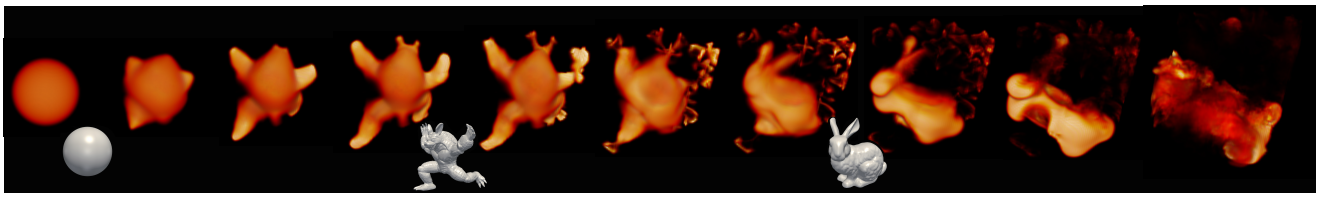

Fig. 10: 3D smoke control example of deforming a sphere first to an armadillo and then to a bunny. This example runs at the resolution of 64^3 with 40 timesteps. The optimization can be accomplished in 7hr.

have genus one and two. Our algorithm can handle such complex cases.

Comparison with LBFGS: We compared our ADMM-based solver with a gradient-based quasi-Newton optimizer in solving the original problem (Equation 5). Specifically, we use LBFGS method [Nocedal and Wright 2006]. Such method approximates the Hessian using a history of gradients calculated by past iterations. We set the history size to be 8, which is typical. We use same stopping criteria for both LBFGS and our method. Under this setting, we compared the performance of LBFGS and the ADMM solver on two of our 2D examples: Figure 1 and Figure 8. For the example of letter matching in Figure 1, LBFGS algorithm takes 4hr and 71 iterations to converge. While for the example of changing regularization in Figure 8, LBFGS algorithm takes 12hr and 152 iterations at $r = 10^2$, 15hr and 170 iterations at $r = 10^3$, and 16hr and 212 iterations at $r = 10^4$. Therefore, our algorithm is approximately an order of magnitude faster than a typical implementation of LBFGS.

The speedup over LBFGS optimizer occurs for two reasons. First, we break the problem up into the AO subproblem and the NSO subproblem, that have sharply different properties. The AO subproblem is nonsmooth while the NSO subproblem is not. In practice, neither our fixed point iteration scheme in Equation 8 nor the LBFGS algorithm can efficiently solve AO to arbitrarily small KKT residual. Without such decomposition, it takes a very long time to solve the overall optimization problem by taking a lot of iterations. The second reason is the use of warm-started FAS solver for the NSO subproblem. Note that LBFGS algorithm not only takes more iterations, but each iteration is also more expensive. This is mainly because of the repeated gradient evaluation in each LBFGS iteration, where each evaluation runs the adjoint method with a cost equivalent to two passes of fluid resimulation.

Comparison with PD Controller: We also compared our method with simple tracker type controllers such as PD controller [Fattal and Lischinski 2004]. To drive the fluid body towards a target keyframe shape using heuristic ghost forces, PD controllers result in much lower overhead in terms of fluid resimulation, as compared to our approach based on optimal controllers. In contrast, optimal controllers provide better flexibility and robust solutions as compared to PD controllers. A PD controller tends to be sensitive to the parameters used for the ghost forces. Moreover, its performance also depends on the use of the compressible control forces (see Figure 14). These compressible forces can potentially eliminate the visually appealing vortical fluid motions. On the other hand, an optimal controller always achieves exact keyframe timing, while such exact timing requires fine-tuning the strength of control forces in a PD controller, as shown in Figure 14. Moreover, with an optimal controller, users can easily balance between the exactness of keyframe matching and the amount of fluid-like behaviors based on a single parameter r (see Figure 8).

Memory Overhead: Since fluid control problems usually have a high memory overhead, we derive here an analytical upper bound of the memory consumption $M(n, d, N)$:

$$M(n,d,N) \sim \left[(n^d) * (1+d) * 2 * 2\right] * \left[1 + \tfrac{1}{2} + \tfrac{1}{4} \cdots\right] * N = 8n^d(1+d)N,$$

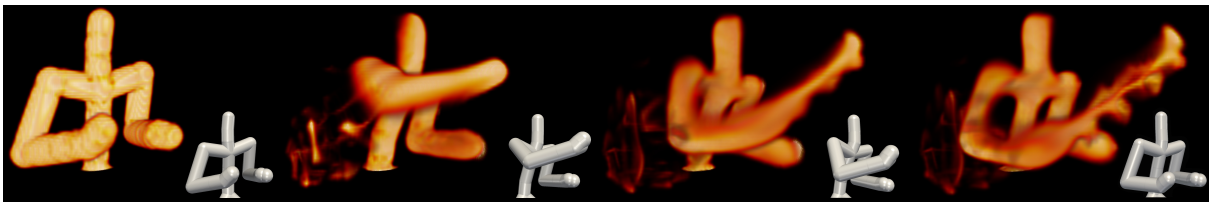

Fig. 11: We generate the famous example of tracking smoke with a dense sequence of keyframes, which comes from human motion capture data. Our algorithm converges and generates rich smoke drags within 5 ADMM iterations.

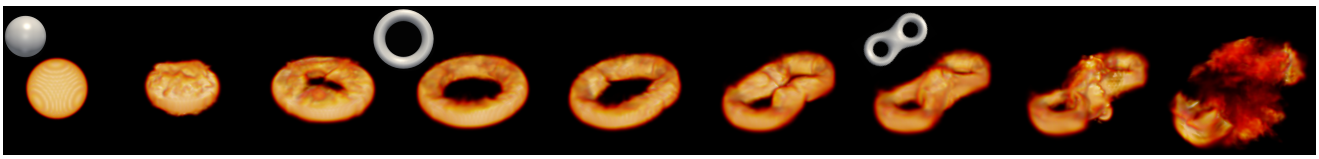

Fig. 12: Example of smoke control where the keyframes have varying genera. The initial frame is a sphere (genus 0). The first keyframe located at frame 20 is a torus (genus 1) and the second keyframe located at frame 40 is the shape eight (genus 2). The resolution is $64^2 \times 32/40$ and the overall optimization takes 5hr with $r = 10^3$.

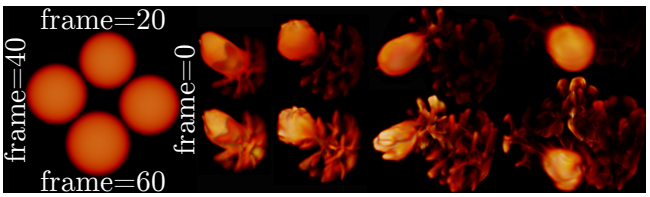

Fig. 13: A moving smoke sphere guided by the 3 keyframes (left). We experimented with $r = 10^3$ (top) and $r = 10^4$ (bottom). Larger regularization results in more wake flow behind moving smoke bodies. The same effect can be observed in Figure 8.

where n is the grid resolution, d is the dimension, and N is the number of timesteps. To derive this bound, note that we can reuse the memory consumed by Algorithm 2 in Algorithm 1, and Algorithm 2 always consumes more memory than Algorithm 1, so that we only consider the memory overhead of Algorithm 2. The first term $n^d * (1 + d)$ is the number of variables needed for storing a pair of pressure and velocity fields. This number is doubled because we need to store $u_i, \bar{p}_i$ in addition to v_i, p_i at each timestep. We double it again because we need additional memory for storing $\mathbf{res}$ in FAS. Finally, the power series is due to the hierarchy of grids. At first observation, this memory overhead is higher than [Treuille et al. 2003; McNamara et al. 2004] since we require additional memory for storing the dual variables at multiple resolutions. However, due to the quasi-Newton method involved in their approach, additional memories are needed to store a set of L gradients to approximate the inverse of the Hessian matrix. L is usually $5 \sim 10$, leading to the following upper bound:

$$M_{LBFGS}(n, d, N) \sim \left[(n^d) * (1 + d)\right] * L * N = Ln^d(1 + d)N.$$

In our benchmarks, the memory overheads of our ADMM and LBFGS solvers are comparable.

Convergence Analysis: Here we analyze the convergence of our approach and discuss some modifications towards improved convergence of Algorithm 3. We have applied some of these modifications for Line 7 and Line 9 of Algorithm 3, which then takes a slightly more complex form.

Fig. 14: We deform a smoke ball into a dragon using our method (top left) and PD controller with different parameter settings. Top Right: Without gathering forces, the keyframe shape is not matched. Bottom Left: The animation appears to be oscillatory without viscous forces, resulting in lots of escaped smokes. Bottom Right: A stable, non-oscillatory animation that matches the keyframe well can be achieve by fine-tuning the three parameters: the strength of guiding forces, the strength of gathering forces, and the strength of viscous forces.

For our AO solver (Line 7 of Algorithm 3), we observe that it can be difficult for Algorithm 1 to converge to an arbitrarily small KKT residual in each loop of Algorithm 3. As illustrated in Algorithm 1, one could use a simple strategy that can guarantee function value decreases by blending a new solution with the previous solution and tuning the blending factor in a way similar to the line search algorithm. This modification has low computational overhead since one does not need to apply the costly solenoidal projection operator

Q again after the blending, as the sum of two solenoidal vector fields is still solenoidal. In our benchmarks, this strategy leads to a convergent algorithm with low overhead, but the error reduction rate after the first few iterations can still be slow.

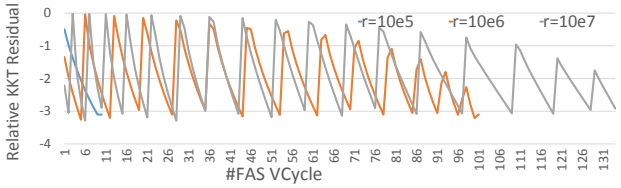

Fig. 15: Convergence history of the NSO solver in the Circle Bunny example. When the regularization coefficient r is extremely large, we have to treat the FAS-Vcycle as a subproblem solver of the LM algorithm and the entire NSO solve requires many more FAS-Vcycles.

The same analysis can also be used for the NSO solver (Line 9 of Algorithm 3). To ensure convergence of Algorithm 2, we could add a perturbation to the penalty coefficient K in the Hessian matrix Equation 10. Note that as $K \to \infty$, $v_i \to v_i^*$. Therefore, this strategy essentially makes Algorithm 2 the subproblem solver for the Levenberg-Marquardt algorithm [Nocedal and Wright 2006], which in turn guarantees convergence. As illustrated in Figure 15, Levenberg-Marquardt modification can be necessary when one uses extremely large regularization r, because we observe that the convergence rate decreases as r increases. In these settings, however, many more FAS-Vcycles are needed to solve the NSO subproblem and the advantage over the LBFGS solver also decreases.

Finally, for the ADMM outer loop (Line 5 of Algorithm 3), state-of-the-art results showing its convergence rely on strong assumptions of its objective function, such as global convexity. Therefore, our current implementation of outer loop is not guaranteed to converge. However, such guarantee can be provided by using a standard Augmented Lagrangian solver, instead of ADMM solver. Specifically, one can run Algorithm 3 without applying Line 18 until the decrease in function value is lower than some threshold. Further exploration of this option is left as future work.

6. CONCLUSION AND LIMITATIONS

In our work, we present a new algorithm for the optimal control of smoke animation. Our algorithm finds the stationary point of the KKT conditions, solving for both primal and dual variables. Our key idea is to refine primal as well as dual variables in a warm-started manner, without requiring them to satisfy the Navier-Stokes equations exactly in each iteration. We tested our approach on several benchmarks and a wide range of parameter choices. The results show that our method can robustly find the locally optimal control forces while achieving an order of magnitude speedup over the gradient-based optimizer, which performs fluid resimulation in each gradient evaluation.

On the downside, our method severely relies on the spatial structure and the staggered grid discretization of the Navier-Stokes equations. This imposes a major restriction to the application of our techniques. Nevertheless, generalizing our idea to other fluid discretization is still possible. For example, our method can be used with a fluid solver discretized on a general tetrahedron mesh such as [Chentanez et al. 2007; Pavlov et al. 2011] since the KKT conditions are invariant under different discretizations, and the three operators to define FAS stay valid. On the other hand, generalizing our method to free-surface flow or to handle internal boundary conditions can be non-trivial. The distance metric C_i in Equation 7 needs to be modified to make it aware of the boundaries, e.g., Euclidean distances should be replaced with Geodesic distances. However, modifying the NSO solver to handle the boundaries can be relatively straightforward. This is because our multigrid formulation is the same as a conventional multigrid formulation in spatial domain, using simple trilinear prolongation and restriction operators. Therefore, existing works on boundary aware multigrid such as [Chentanez and Müller 2011] can also be applied to our space-time formulation.

In addition, unlike [Treuille et al. 2003; McNamara et al. 2004], which use a set of template ghost force bases to reduce the search space, our method allows every velocity component to be optimized. This choice is application dependent. For matching smoke to detailed keyframes with lots of high frequency features, our formulation can be useful. However, using a reduced set of template ghost forces could help to avoid the popping artifacts illustrated in Figure 4, and at the same time it allows more user control over the applied control force patterns. For example, the use of vortex force templates encourages more swirly motions in the controlled animations. Moreover, from Figure 8, we can see many small-scale escaping smoke parcels using a large r. Our controller does not apply control forces on these parcels in order to reduce the magnitude of control forces. If these smoke parcels are undesirable in the final animation, a template-based formulation can be used. Combining the control force templates with our formulation is considered as future work.

In terms of computational overhead, since our optimal controller always solves the spacetime optimization by considering all the timesteps, it is much slower than a simple PD controller which considers one timestep at a time. For example, it took more than 26 hours to generate our longest animation with 150 timesteps illustrated in Figure 16. In order to reduce runtime cost, we can use a larger timestep size to reduce the number of timesteps. Fluid simulation with a large timestep size has been addressed in prior work, such as [Lentine et al. 2012]. We handle the large timestep size by using a novel advection operator (Equation 4) with an adaptive order of Taylor expansion. An alternative solution is to use a conventional advection scheme with an adaptive timestep size determined using the CFL condition. A thorough analysis of alternative advection schemes is a good topic of future work. Also, we can lower the spatial resolution in the control phase and then use smoke upsampling methods such as [Nielsen and Bridson 2011] to generate a high quality animation.

Further accelerations to our method are still possible. For example, one can parallelize our algorithm in a distributed environment. Indeed, multigrid is known as one of the most cluster-friendly algorithms. Moreover, meta-algorithms such as multiple shooting [Bock and Plitt 1984] try to break the spacetime optimization into a series of sub-optimizations that consider only a short animation segment and are thus faster to solve. Finally, the benefits of both optimal and PD controllers can be combined by borrowing the idea of receding horizon control [Mayne and Michalska 1990]. In these controllers, optimal control is applied only to a short window of timesteps starting from the current one, and the window keeps being shifted forward to cover the whole animation.

Fig. 16: The same example as Figure 9 in 3D. We use 150 timesteps with keyframes at timestep 50, 100, 150, respectively. Our algorithm converges after 20 iterations and the total computation time is 26 hours.

APPENDIX

A. KKT SYSTEM OF THE NSO SUBPROBLEM

We derive here the KKT system for the NSO subproblem. Instead of simply introducing the Lagrangian multipliers and following standard techniques as we did for the AO subproblem, we present a derivation based on the analysis of the ghost force u_i. We first eliminate the Navier-Stokes constraints by writing u_i as a function of v_i and v_{i+1}. Next, we plug this function into our objective to obtain:

$$\frac{r}{2}\sum_{i=0}^{N-1}\|u_i(v_i,v_{i+1})\|^2+\frac{K}{2}\sum_{i=0}^{N-1}\|v_i-v_i^*\|^2.$$

Taking the derivative of this objective against v_i and considering the additional solenoidal constraints on v_i, we get the first two equations in f:

$$\begin{aligned}&\frac{K}{r}(v_i-v_i^*)+\frac{\partial u_i}{\partial v_i}^T u_i+\frac{\partial u_{i-1}}{\partial v_i}^T u_{i-1}+\nabla\bar{p}_i=0\\&\nabla\cdot v_i=0,\end{aligned}$$

where $\bar{p}_i$ is the Lagrangian multiplier. Now in order to derive the other two conditions in Equation 9, we need to determine the additional pressure p_i. We assert that p_{i+1} is the Lagrangian multiplier of the solenoidal constraints on u_i. In fact, if u_i is not divergence-free, we can always perform a pressure projection on u_i by minimizing $\|u_i-\nabla p_{i+1}\|^2$ to get a smaller objective function value. As a result, u_i must be divergence-free at the optima with p_{i+1} being the Lagrangian multiplier, and we get the two additional equations of f:

$$\begin{aligned}&\frac{v_{i+1}-v_i}{\Delta t}+\mathbf{Adv}\left[v_{i+1}\right]-u_i+\nabla p_{i+1}=0\\&\nabla\cdot u_i=0.\end{aligned}$$

From these two conditions, we can see that $\frac{\partial u_i}{\partial v_i}=-\mathbf{Q}\frac{I}{\Delta t}$, $\frac{\partial u_{i-1}}{\partial v_i}=\mathbf{Q}(\frac{I}{\Delta t}+\frac{\partial\mathbf{Adv}[v_i]}{\partial v_i})$. Here $\mathbf{Q}$ is the solenoidal projection operator introduced in Equation 8. However, we can drop this $\mathbf{Q}$ because we have $\frac{\partial u_{i-1}}{\partial v_i}^T u_i=(\frac{I}{\Delta t}+\frac{\partial\mathbf{Adv}[v_i]}{\partial v_i})^T\mathbf{Q}^T u_i$ and $\mathbf{Q}^T u_i=\mathbf{Q}u_i=u_i$ by the fact that u_i is already solenoidal.

REFERENCES

Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. 1984. Pyramid methods in image processing. *RCA engineer* 29, 6 (1984), 33–41.

Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. 2006. A Controllable, Fast and Stable Basis for Vortex Based Smoke Simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*. Eurographics Association, 25–32.

Hans Georg Bock and Karl-Josef Plitt. 1984. A multiple shooting algorithm for direct solution of optimal control problems. Proceedings of the IFAC World Congress.

Alfio Borzi and R Griesse. 2005. Experiences with a space–time multigrid method for the optimal control of a chemical turbulence model. *International journal for numerical methods in fluids* 47, 8-9 (2005), 879–885.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3, 1 (2011), 1–122.

Achi Brandt and Oren E Livne. 2011. *Multigrid techniques: 1984 guide with applications to fluid dynamics*. Vol. 67. SIAM.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* 16, 5 (1995), 1190–1208.

Nuttapong Chentanez, Bryan E Feldman, François Labelle, James F O'Brien, and Jonathan R Shewchuk. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 219–228.

Nuttapong Chentanez and Matthias Müller. 2011. Real-time eulerian water simulation using a restricted tall cell grid. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 82.

Raanan Fattal and Dani Lischinski. 2004. Target-driven smoke animation. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 441–448.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 15–22.

James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. 2014. From Capture to Simulation: Connecting Forward and Inverse Problems in Fluids. *ACM Trans. Graph.* 33, 4, Article 139 (July 2014), 11 pages. DOI:http://dx.doi.org/10.1145/2601097.2601147

F. H. Harlow and J. E. Welch. 1965. Numerical calculation of time-dependent viscous incompressible flowof fluid with free surfaces. *Physics of Fluids* 8 (1965), 2182–2188.

Roland Herzog and Karl Kunisch. 2010. Algorithms for PDE-constrained optimization. *GAMM-Mitteilungen* 33, 2 (2010), 163–176.

Michael Hinze, Michael Köster, and Stefan Turek. 2012. A space-time multigrid method for optimal flow control. In *Constrained optimization and optimal control for partial differential equations*. Springer, 147–170.

Michael Lentine, Matthew Cong, Saket Patkar, and Ronald Fedkiw. 2012. Simulating Free Surface Flow with Very Large Time Steps. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 107–116. http://dl.acm.org/citation.cfm?id=2422356.2422373

Brian P Leonard. 1979. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer methods in applied mechanics and engineering* 19, 1 (1979), 59–98.

David Q Mayne and Hannah Michalska. 1990. Receding horizon control of nonlinear systems. *Automatic Control, IEEE Transactions on* 35, 7 (1990), 814–824.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. In *ACM Transactions On Graphics (TOG)*, Vol. 23. ACM, 449–456.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 43.

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 21–28. http://dl.acm.org/citation.cfm?id=2982818.2982822

Michael B Nielsen and Robert Bridson. 2011. Guide shapes for high resolution naturalistic liquid simulation. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 83.

Michael B Nielsen and Brian B Christensen. 2010. Improved variational guiding of smoke animations. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 705–712.

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive localized liquid motion editing. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 184.

Dmitry Pavlov, Patrick Mullen, Yiying Tong, Eva Kanso, Jerrold E Marsden, and Mathieu Desbrun. 2011. Structure-preserving discretization of incompressible fluids. *Physica D: Nonlinear Phenomena* 240, 6 (2011), 443–458.

Nick Rasmussen, Doug Enright, Duc Nguyen, Sebastian Marino, Nigel Sumner, Willi Geiger, Samir Hoon, and Ron Fedkiw. 2004. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 193–202.

Karthik Raveendran, Nils Thuerey, Chris Wojtan, and Greg Turk. 2012. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 255–264.

Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. 2014. Blending liquids. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 137.

Lin Shi and Yizhou Yu. 2005. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 229–236.

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics (TOG)* 22 (2003), 716–723.

SP Vanka. 1983. *Fully coupled calculation of fluid flows with limited use of computer storage*. Technical Report. Argonne National Lab., IL (USA).

Xinxin Zhang and Robert Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 206.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 965–972.