Fluid Control with Laplacian Eigenfunctions

Yixin Chen
University of Toronto
Toronto, ON, Canada
yixin@cs.toronto.edu

David I.W. Levin
University of Toronto
Toronto, ON, Canada
Nvidia
Toronto, ON, Canada
diwlevin@cs.toronto.edu

Timothy R. Langlois
Adobe Research
Seattle, WA, USA
tlangloi@adobe.com

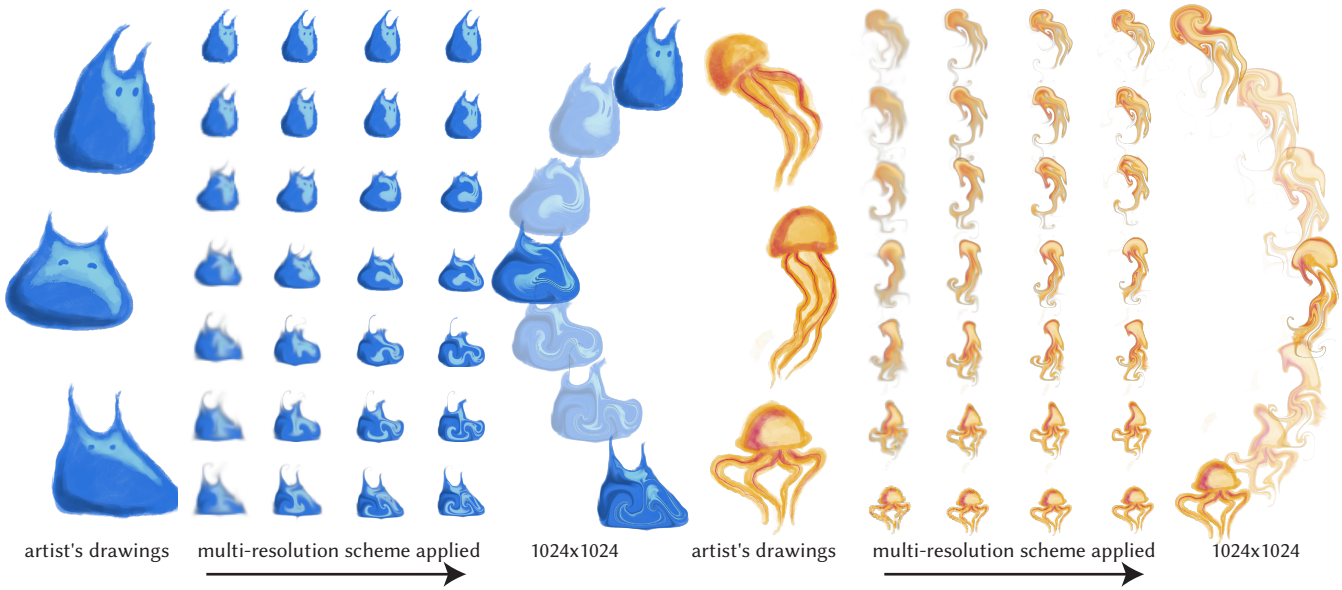

Figure 1: Using several artist's drawings as keyframes, we efficiently optimize fluid simulations which match them. We can achieve high resolution results an order-of-magnitude faster than previous methods. We run our optimization at a sequence of resolutions (from 64×64 to 1024×1024), with warm starting between. This aids convergence and avoids gradient collapse of the objective function if keyframes are too dissimilar.

ABSTRACT

Physics-based fluid control has long been a challenging problem in balancing efficiency and accuracy. We introduce a novel physics-based fluid control pipeline using Laplacian Eigenfluids. Utilizing the adjoint method with our provided analytical gradient expressions, the derivative computation of the control problem is efficient and easy to formulate. We demonstrate that our method is fast enough to support real-time fluid simulation, editing, control, and optimal animation generation. Our pipeline naturally supports multi-resolution and frequency control of fluid simulations. The effectiveness and efficiency of our fluid control pipeline are validated through a variety of 2D examples and comparisons.

SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657468

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**.

KEYWORDS

physics-based animation, fluid control, optimization, adjoint method

ACM Reference Format:
Yixin Chen, David I.W. Levin, and Timothy R. Langlois. 2024. Fluid Control with Laplacian Eigenfunctions. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657468

1 INTRODUCTION

Fluids have long been an integral part of computer graphics. Visual storytelling is often enhanced with puffs of smoke, splashing water, and explosive flames. Simulation methods have improved dramatically, enabling more complex effects, as evidenced by works such as Pixar's *Elemental*, which contains fluid simulation in almost every frame. However, control of fluid simulations remains a challenging

problem. Setting simulation parameters and initialization to match artistic intent can be time-consuming. Getting just enough, but not too much, water to splash out of a cup can require many iterations of a simulation. In some cases, users want outputs that feel like fluids but are nonphysical and require non-intuitive control forces to get a simulation to match.

Numerous methods have been explored to control fluids. We discuss these in detail in §2.2, but they all struggle with the difficulty of fluid control in general. The evolution of fluids is highly nonlinear, with a large number of degrees of freedom, resulting in a formidable optimization challenge. Previous methods are mostly too slow to use in real-time applications. Moreover, the artistic preference for fluids to exhibit nonphysical behaviors for emphasis and effect further exacerbates the difficulty of achieving effective fluid control.

We demonstrate that the method of Laplacian Eigenfluids [De Witt et al. 2012; Liu et al. 2015; Cui et al. 2018] is amenable to control and offers several distinct advantages. The method represents the velocity field as a weighted sum of laplacian eigenfunctions, where the weight vector is integrated over time, greatly reducing the number of DOFs in the simulation. It is intrinsically divergence-free, so there is no divergent mode leakage, which can happen with standard numerical pressure projection. The eigenfunction representation naturally provides frequency control, and the velocity fields are spatially smooth, which removes the need for smoothness regularization. When objectives are defined solely with velocity-based keyframes, optimization is extremely fast as it can be conducted entirely within the reduced space. This approach provides a distinct separation of velocity DOFs and density/image resolution, thereby allowing for efficient control.

As is common in unconstrained optimization fluid control, users can optimize for control forces to satisfy specified keyframes. Additionally, we show that the speed of Eigenfluids enables real-time response, allowing users to iteratively "sculpt" fluid animations either by modifying keyframes or by adjusting pathlines. We provide analytical expressions for the necessary gradients and show that our method can produce results comparable to previous methods with an order-of-magnitude speedup.

2 RELATED WORK

2.1 Fluid simulation

The field of fluid simulation in computer graphics has experienced substantial progress over decades. Bridson [2015] provides a good overview. We utilize Laplacian Eigenfluids [De Witt et al. 2012; Liu et al. 2015; Cui et al. 2018] to solve the incompressible Navier-Stokes equations using a reduced space to represent the velocity field. This provides a separation between the velocity DOFs and the density field DOFs, which is often desirable: lower-frequency deformation can be pleasing when applied to high-resolution artwork. It is very fast and intrinsically divergence-free, making it well suited to control.

2.2 Fluid control

Starting from [Foster and Metaxas 1997], one of the first controllers for fluids, fluid control in computer graphics has evolved significantly over the past two decades.

2.2.1 Optimization-based control. Optimization-based fluid control has emerged as a powerful tool for directing fluid simulations, providing systematic and efficient means to achieve desired behaviors. The optimization process facilitates the exploration of control strategies, such as the minimization of certain energy functions or adherence to specific constraints, resulting in simulations that closely align with user-defined goals.

Keyframes are a popular control method, with seminal work [Treuille et al. 2003] using keyframes to direct smoke simulations. This method defined an objective function to measure the difference between the current state of the fluid and the target state, provided derivatives of each fluid operation, and used the L-BFGS method to find optimal control forces. However, it is prohibitively expensive to compute the derivatives, only allowing low-dimensional control forces to be applied. It was refined in [McNamara et al. 2004], which employed the adjoint method to compute the derivatives more efficiently. The problem was reformulated as a constrained optimization [Pan and Manocha 2017; Inglis et al. 2017], which, with the application of the primal-dual algorithm, further improved performance. More recently, Tang et al. [2021] improved performance through frequency-aware force field reduction.

Beyond keyframe-based methods, various other optimization-based control techniques have been explored. A series of papers from Nielsen et al. [2009; 2010; 2011] used lower resolution simulation to control high-resolution smoke and liquid animations. Predefined patterns [Yuan et al. 2011] and meshes [Raveendran et al. 2012] are used to manipulate fluids. Gregson et al. [2014] captured real-world fluid behaviors and translated them into simulations. Raveendran et al. [2014] proposed a technique for blending multiple liquid simulations. Several flow interpolation techniques [Thuerey 2016; Sato et al. 2018a] have been presented for smoother and more controllable results. Later, Eckert et al. [2018] discussed a method for integrating fluid density and motion data from single-view inputs to enhance simulation realism, while Flynn et al. [2019] offered a solution for intelligently resizing fluid simulation data. Takahashi and Lin [2019] explored the transfer of parameters from real-world video footage to virtual fluid animations. Sato et al. [2018b] demonstrated a technique for transferring turbulence styles between simulations. However, achieving real-time performance and high-resolution control remains an ongoing challenge.

Recent approaches have leveraged deep learning. Schenck and Fox [2018] integrated differentiable fluid dynamics into neural networks, enabling more accurate modeling of fluid behavior. Kim et al. [2019] used neural networks for style transfer in smoke simulations. Holl et al. [2020] developed a method to control partial differential equations (PDEs) through differentiable physics, enhancing the ability of neural networks to predict and manage complex physical systems. Chu et al. [2021] presented a data-driven conditional adversarial model that generated plausible velocity fields from a single frame of a density field, while a two-stage generative model was proposed by Xie et al. [2022] that assisted in the creation of smoke illustrations from sketches. Kim et al. [2022] explored a deep learning approach for reconstructing detailed 3D smoke densities from simple artist sketches. Aurand et al. [2022] proposed an efficient technique for applying neural style transfer to volumetric simulations. Tang et al. [2023] introduced a physics-informed neural

corrector that improves the control and accuracy of fluid simulations, particularly in scenarios involving complex deformations. Deep learning methods can have trouble extrapolating to novel flows outside of their training data.

2.2.2 Optimization-free control. In contrast to optimization-based approaches, several works have explored direct control. Fattal and Lischinski [2004] introduced forcing terms to direct smoke animations towards targets without requiring optimization. Other optimization-free fluid control schemes have employed geometric potentials [Hong and Kim 2004; Shi and Yu 2005b], advected radial basis functions [Pighin et al. 2004], guiding objects [Shi and Yu 2005a], and scale-dependent forces [Thürey et al. 2009]. Mihalef et al. [2004] presented a method for realistically animating and controlling the complex dynamics of breaking waves, capturing both their visual and physical properties.

Various feature-based [Schpok et al. 2005] fluid control techniques have been explored, including path-based [Kim et al. 2006], vortex-based [Angelidis et al. 2006], filament-based [Weißmann and Pinkall 2010], particle-based [Rasmussen et al. 2004; Madill and Mould 2013], and preview-based sampling [Huang et al. 2011]. Yang et al. [2013] introduced a unified approach to smoke control using signed distance fields. Ren et al. [2013] explored techniques for analyzing and modulating the intrinsic multi-scale features in fluid simulations. Manteaux et al. [2016] introduced a space-time framework for sculpting liquid animations, allowing for seamlessly editing pre-computed animations of liquid.

2.2.3 Interactive control. While interactive fluid simulation methods have seen significant advancements, real-time interactive fluid control problems are largely unexplored in computer graphics, where the challenges mostly lie in the need for integration of user input and real-time responsiveness. Pan et al. [2013] introduced an interactive system for real-time editing of localized liquid motions using geometric deformation. Yan et al. [2020] presented a novel system to synthesize realistic liquid splashes from simple user sketches, which utilizes a conditional generative adversarial network (cGAN) trained with physics-based simulation data. Schoentgen et al. [2020] used precomputed templates to direct particle-based simulations. Advancing further, Tang et al. [2023] employed convolutional neural networks trained with physics-inspired loss functions along with a differentiable fluid simulator to provide an efficient workflow for rectifying deformed fluid flows. However, these methods generally focus on localized time spans (static in some cases), and sketch control is not fine-grained.

3 FLUID CONTROL

We start with a short overview of the Laplacian Eigenfluids method. We then describe the design of our objective function, which incorporates standard keyframe control as well as control with pathlines and obstacles. The speed of Eigenfluids enables interactive user guidance of the optimizer. We also describe how the method easily enables multi-resolution and frequency-based control.

Notation. Uppercase bold denotes matrices, while lowercase bold is used for vectors. We use superscripts to denote a value at a particular time step or to index instances of vectors. Subscripts are used to index entries or segments of vectors/matrices/tensors.

3.1 Laplacian Eigenfluids

A full description is provided in [De Witt et al. 2012; Cui et al. 2018]. To solve the incompressible Navier-Stokes equations, the fluid velocity field $\boldsymbol{u} \in \mathbb{R}^{md}$ on a grid of m cells in dimension d is represented as a combination of r basis functions $\boldsymbol{u} = \sum_{k=1}^{r} w_k \boldsymbol{\Psi}^k = \boldsymbol{U}\boldsymbol{w}$, where $\boldsymbol{U}$ denotes the transformation matrix between the two representations.[1] The basis functions are eigenfunctions of the vector laplacian operator, so are intrinsically divergence-free. In this basis representation, the change of the velocity field can be described by the update of the weight vector $\boldsymbol{w}$, using its time derivative

$$\dot{\boldsymbol{w}}_g = \sum_{h=1}^{r}\sum_{i=1}^{r} w_h w_i \mathbb{C}_{ghi} \tag{1}$$

where the entries of the 3rd-order advection tensor are

$$\mathbb{C}_{ghi} = \int_{\Omega} \left(\nabla \times \boldsymbol{\Psi}^i\right) \cdot \left(\boldsymbol{\Psi}^g \times \boldsymbol{\Psi}^h\right) d\Omega. \tag{2}$$

The weight vector is advanced using an implicit trapezoidal update, with an operator splitting scheme to apply damping

$$\tilde{\boldsymbol{w}}^{t+1} = \left(\frac{\Delta t}{2}\tilde{\boldsymbol{C}}^{t+1}\tilde{\boldsymbol{w}}^{t+1} + \frac{\Delta t}{2}\boldsymbol{C}^t\boldsymbol{w}^t + \boldsymbol{w}^t + \boldsymbol{f}^t\right) \tag{3}$$

$$\boldsymbol{w}^{t+1} = \boldsymbol{D}\tilde{\boldsymbol{w}}^{t+1} \tag{4}$$

where the notation $\boldsymbol{C}^t = \mathbb{C} \times_3 \boldsymbol{w}^t = \sum_{i=1}^{r} w_i^t \mathbb{C}_{ghi}$ denotes contraction over the 3rd dimension to produce a matrix, $\boldsymbol{f}^t$ represents any external forces that are applied, and $\boldsymbol{D} = e^{\nu\Delta t\boldsymbol{\Lambda}}$ is the diagonal damping matrix ($\boldsymbol{\Lambda}$ is a diagonal matrix of the basis eigenvalues). In practice, just a single Newton iteration is used, and conjugate gradients on the normal form work well. Typically, $r \ll md$, so this method is very efficient. For advecting a density field $\boldsymbol{\rho}$ through the velocity, the full space velocity $\boldsymbol{u}$ is formed, and a standard advection scheme is used. This is done efficiently using inverse sine and cosine transforms (i.e., the matrix $\boldsymbol{U}$ is never explicitly formed).

3.2 Optimization problem

Control is posed as a minimization problem, where we aim to find a sequence of control forces $\boldsymbol{f}$ which minimize an objective function φ that incorporates desired states/keyframes. Without a superscript, vectors such as the control force vector $\boldsymbol{f} \in \mathbb{R}^{nr} = [\boldsymbol{f}^{0\top}, \boldsymbol{f}^{1\top}, \ldots]^\top$ represent the concatenation of vectors from all n timesteps. We optimize

$$\arg\min_{\boldsymbol{f}} \varphi(\boldsymbol{f}, \boldsymbol{q}) \tag{5}$$

where $\boldsymbol{q}^t = [\boldsymbol{w}^{t\top}, \boldsymbol{\rho}^{t\top}]^\top$ is the state of the system at time t. The objective function of standard keyframe control is defined as

$$\varphi(\boldsymbol{f}, \boldsymbol{q}) = \underbrace{k_s \sum_{t=0}^{n} \left\|\boldsymbol{f}^t\right\|^2}_{\xi_s} + \underbrace{k_w \sum_{t\in\boldsymbol{K}_w} \left\|\boldsymbol{w}^t - \boldsymbol{w}^{t*}\right\|^2}_{\xi_w} + \underbrace{k_d \sum_{t\in\boldsymbol{K}_d} \left\|\boldsymbol{\rho}^t - \boldsymbol{\rho}^{t*}\right\|^2}_{\xi_d},$$

consisting of three terms: ξ_s for minimizing control forces, ξ_w for controlling velocity and ξ_d for controlling density, respectively. k_s, k_w, and k_d are all scaling parameters. $\boldsymbol{K}_w$ and $\boldsymbol{K}_d$ are sets of keyframe times for velocity and density. $\boldsymbol{w}^{t*}$ and $\boldsymbol{\rho}^{t*}$ represent

[1] Each basis function is associated with a wave vector $\boldsymbol{k} \in \mathbb{R}^d$. We use the notation Ψ^k to generically iterate over the basis functions.

the keyframes. The gradients required for this optimization are provided in §4. To satisfy the incompressibility constraint, we normalize all input keyframes to have the same amount of total density.

3.2.1 Obstacles. Obstacles are usually handled by a simple projection method in the Eigenfluids framework. At every step, the full-space velocity is reconstructed, and it is adjusted to enforce boundary conditions, and then the velocity is transformed back to the reduced space.

While this method could easily be incorporated into our simulation and optimization, we found that adapting the objective function to handle obstacles allows the object constraints to be treated in a soft way, balancing them against the other objectives. Specifically, we define a function

$$\tilde{\boldsymbol{u}}_{\boldsymbol{x}} = \begin{cases} 0 & \boldsymbol{x} \text{ is not in an obstacle} \\ \hat{\boldsymbol{n}} & \boldsymbol{x} \text{ is on the boundary of an obstacle} \\ \boldsymbol{u}_{\boldsymbol{x}} & \boldsymbol{x} \text{ is inside of an object} \end{cases} \tag{6}$$

where $\boldsymbol{x}$ denotes the position of grid cells and $\hat{\boldsymbol{n}}$ is the outward normal of the obstacle at position $\boldsymbol{x}$. Then we can extend our objective function with a penalty term ξ_o to effectively handle obstacles, where

$$\xi_o = k_o \sum_{\boldsymbol{x}} \left(\boldsymbol{u}_{\boldsymbol{x}} \cdot \tilde{\boldsymbol{u}}_{\boldsymbol{x}}\right)^2 . \tag{7}$$

3.2.2 Pathlines. The pathlines of a fluid can be useful for control, as they give users a visualization of how portions of the fluid move over time. These can be easily computed, visualized, and controlled as the paths of massless particles in the fluid flow (Fig. 2). To add a pathline control, the user can interactively add a particle and a desired forward or backward path for it. We then augment our system state $\boldsymbol{q}^t = [\boldsymbol{w}^{t\top}, \boldsymbol{\rho}^{t\top}, \boldsymbol{p}^{t\top}]^\top$ with the particle position $\boldsymbol{p}^t$. The particle is integrated through the fluid flow, and we enhance our cost function by incorporating a term

$$\xi_p = k_p \sum_{t \in \boldsymbol{K}_p} \left\| \boldsymbol{p}^t - \boldsymbol{p}^{t*} \right\|^2 . \tag{8}$$

The fluid flow will be then adjusted to match the specified pathlines.

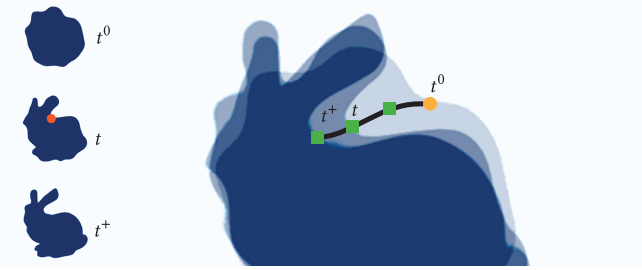

Figure 2: During user interaction, pathlines can be used to look forward or backward in time. When selecting a spatial point at time t, the user can choose to trace the pathline backwards to the start t^0 and forwards to a time t^+. The user can then specify any of the pathline's positions at times after t^0. Multiple pathlines could be used at once, each of different lengths.

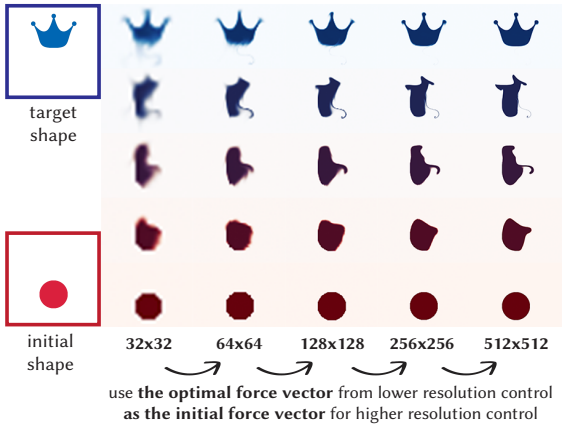

Figure 3: In our multi-resolution scheme, we downsample density keyframes, run the optimizer to convergence, then use that solution to initialize an optimization at the next resolution level.

3.3 Multi-resolution pyramid

When keyframes are very far from the current state, the L2 cost function can break down, giving zero gradients. To avoid this, we optimize on a sequence of density resolutions. Illustrated in Fig. 3, we first optimize on a low-resolution image until convergence, then use that solution to initialize the optimization at the next level. We note that the velocity DOFs in the reduced space do not change; they are just expressed on different resolution grids (through different resolution DCTs). The multi-resolution pyramid allows us to control the degree to which we match keyframes. We can match closely using high-resolution velocities or use low-resolution velocities to leave room for stylistic smoky elements.

3.4 Frequency control

Due to the high non-linearity of the problem, there are many local minima. As observed previously [Tang et al. 2021], controlling the frequency content of control forces can aid the optimizer. Laplacian Eigenfluids naturally supports this through reweighting of the advection tensor. We use a cascade of ideal low-pass filters, which amounts to limiting the number of basis functions. As opposed to just filtering the frequency content of the control forces at different stages of optimization, we actually reduce the DOFs of the system, making optimization much faster at lower frequencies.

Beyond aiding the optimizer, having separate controls for the velocity and density resolutions can be a useful tool for users. As observed previously [Muller et al. 2004], low-dimensional dynamics can often look appealing when applied to high-resolution textures (Fig. 4). Eigenfluids provides this for fluids, as the velocity/forces and density are represented in different spaces.

4 GRADIENT COMPUTATION

Here we provide an overview of how the objective cost function gradient is computed and provide derivations for the non-trivial components.

4.1 Adjoint Method

For optimization, we need the derivative with respect to the control forces:

$$\frac{d\varphi}{df} = \frac{\partial\varphi}{\partial \boldsymbol{q}}\frac{d\boldsymbol{q}}{df} + \frac{\partial\varphi}{\partial f} \tag{9}$$

which can be evaluated efficiently using the adjoint method as described by McNamera et al. [2004]. This is akin to reverse-mode autodiff with several custom gradient operators, but it only requires storing the simulation states, not the entire compute graph.

Specifically, we define a stepping function $\boldsymbol{s}$ and we can represent the update of state at each time step as

$$\boldsymbol{q}^{t+1} = \boldsymbol{s}^t(\boldsymbol{q}^t, f) \qquad \Longrightarrow \qquad \boldsymbol{q} = \boldsymbol{s}(\boldsymbol{q}, f). \tag{10}$$

Then the overall gradient can be written as

$$\frac{d\varphi}{df} = \boldsymbol{r}^\top \frac{\partial \boldsymbol{s}}{\partial f} + \frac{\partial\varphi}{\partial f} \tag{11}$$

$$= \boldsymbol{r}^\top \begin{pmatrix} \frac{\partial \boldsymbol{w}^1}{\partial f^0} & \boldsymbol{0} & \boldsymbol{0} & \dots \\ \frac{\partial \boldsymbol{\rho}^1}{\partial f^0} & \boldsymbol{0} & \boldsymbol{0} & \dots \\ \boldsymbol{0} & \frac{\partial \boldsymbol{w}^2}{\partial f^1} & \boldsymbol{0} & \dots \\ \boldsymbol{0} & \frac{\partial \boldsymbol{\rho}^2}{\partial f^1} & \boldsymbol{0} & \dots \\ \boldsymbol{0} & \boldsymbol{0} & \ddots & \\ \vdots & \vdots & & \end{pmatrix} + \frac{\partial\varphi}{\partial f} \tag{12}$$

Where $\boldsymbol{r}^n = (\partial\varphi/\partial \boldsymbol{q}^n)^\top$, and

$$\boldsymbol{r}^t = \left(\frac{\partial \boldsymbol{s}^t}{\partial \boldsymbol{q}^t}\right)^\top \boldsymbol{r}^{t+1} + \left(\frac{\partial\varphi}{\partial \boldsymbol{q}^t}\right)^\top \tag{13}$$

$$= \begin{pmatrix} \frac{\partial \boldsymbol{w}^{t+1}}{\partial \boldsymbol{w}^t} & \frac{\partial \boldsymbol{w}^{t+1}}{\partial \boldsymbol{\rho}^t} \\ \frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{w}^t} & \frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{\rho}^t} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{r}_w^{t+1} \\ \boldsymbol{r}_\rho^{t+1} \end{pmatrix} + \begin{pmatrix} \frac{\partial\varphi}{\partial \boldsymbol{w}^t} & \frac{\partial\varphi}{\partial \boldsymbol{\rho}^t} \end{pmatrix}^\top \tag{14}$$

In practice, to compute the gradient, we first run a forward simulation and store all the states. We then use (14) to compute the $\boldsymbol{r}$ vector and (12) to compute the gradient.

4.2 Velocity derivatives

Here we compute the $\frac{\partial \boldsymbol{w}^{t+1}}{\partial \boldsymbol{w}^t} = \frac{\partial \boldsymbol{w}^{t+1}}{\partial \tilde{\boldsymbol{w}}^{t+1}}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t} = \boldsymbol{D}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t}$ part of (14), and also the $\frac{\partial \boldsymbol{w}^{t+1}}{\partial f^t} = \boldsymbol{D}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t}$ terms from (12). The velocity field doesn't depend on the advected scalar, so $\frac{\partial \boldsymbol{w}^{t+1}}{\partial \boldsymbol{\rho}^t} = 0$.

Taking a derivative of (3) with respect to $\boldsymbol{w}^t$ gives

$$\underbrace{\left(\boldsymbol{I} - \frac{\Delta t}{2}\left[\mathbb{C}\times_2 \tilde{\boldsymbol{w}}^{t+1} + \mathbb{C}\times_3 \tilde{\boldsymbol{w}}^{t+1}\right]\right)}_{\boldsymbol{A}^{t+1}} \underbrace{\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t}}_{\boldsymbol{X}^{t+1}} = \underbrace{\frac{\Delta t}{2}\left(\mathbb{C}\times_2 \boldsymbol{w}^t + \mathbb{C}\times_3 \boldsymbol{w}^t\right) + \boldsymbol{I}}_{\boldsymbol{B}^t} \tag{15}$$

Taking the derivative with respect to $\boldsymbol{f}^t$ gives

$$\underbrace{\left(\boldsymbol{I} - \frac{\Delta t}{2}\left[\mathbb{C}\times_2 \tilde{\boldsymbol{w}}^{t+1} + \mathbb{C}\times_3 \tilde{\boldsymbol{w}}^{t+1}\right]\right)}_{\boldsymbol{A}^{t+1}} \underbrace{\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial f^t}}_{\boldsymbol{Y}^{t+1}} = \boldsymbol{I} \tag{16}$$

But these are expensive to evaluate directly, as they require matrix solves against large matrices. We can again use the dual/adjoint method.

4.2.1 Dual Computation. Referring to equations (12) and (14), we do not need the full derivative matrices; we need to compute

$$\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t}^\top \boldsymbol{D}\boldsymbol{r}^{t+1} = \left(\boldsymbol{r}^{t+1\top}\boldsymbol{D}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t}\right)^\top \quad \text{such that} \quad \boldsymbol{A}^{t+1}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t} = \boldsymbol{B}^t \tag{17}$$

and

$$\boldsymbol{r}^{t\top}\boldsymbol{D}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial f^t} \quad \text{such that} \quad \boldsymbol{A}^{t+1}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial f^t} = \boldsymbol{I} \tag{18}$$

This is two applications of the dual method. To compute (17), compute

$$\boldsymbol{A}^{t+1\top}\boldsymbol{c}^{t+1} = \boldsymbol{D}\boldsymbol{r}^{t+1} \qquad \Longrightarrow \qquad \boldsymbol{c}^{t+1\top}\boldsymbol{B}^t = \boldsymbol{r}^{t+1\top}\boldsymbol{D}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial \boldsymbol{w}^t} \tag{19}$$

and to compute (18), compute

$$\boldsymbol{A}^{t+1\top}\boldsymbol{d}^{t+1} = \boldsymbol{D}\boldsymbol{r}^t \qquad \Longrightarrow \qquad \boldsymbol{d}^{t+1\top} = \boldsymbol{r}^{t\top}\boldsymbol{D}\frac{\partial \tilde{\boldsymbol{w}}^{t+1}}{\partial f^t} \tag{20}$$

This process requires two extra matrix solves, two extra matrix vector products, and one extra matrix contraction compared to the forward simulation (we already have the contraction over the third dimension from the forward simulation). The $\boldsymbol{A}$ matrices lose the skew-symmetry property of those in (3), due to the contraction over the second dimension. Also note that in the first Newton iteration, we initialize $\tilde{\boldsymbol{w}} = \boldsymbol{w}^t$, so $\boldsymbol{B}^t = 2\boldsymbol{I} - \boldsymbol{A}^{t+1}$.

4.3 Density derivatives

Here we compute the $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{w}^t} = \frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{u}^t}\frac{\partial \boldsymbol{u}^t}{\partial \boldsymbol{w}^t}$ and $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{\rho}^t}$ parts of (14). The $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial f^t}$ terms of (12) are all zero (because $\boldsymbol{s}^t$ only depends on $\boldsymbol{q}^t$).

As for density advection, we use semi-lagrangian advection. The derivatives $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{u}^t}$ and $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{\rho}^t}$ are provided by [McNamara et al. 2004]. The second term is straightforward to compute. $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{w}^t}$ is tricky, because $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{u}^t}$ is a large, sparse matrix, and $\frac{\partial \boldsymbol{u}^t}{\partial \boldsymbol{w}^t}$ is a DST/DCT transform. But note that during the adjoint computation, what we need to compute is

$$\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{w}^t}^\top \boldsymbol{r}_\rho^{t+1} = \left(\boldsymbol{r}_\rho^{t+1\top}\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{w}^t}\right)^\top = \left(\boldsymbol{r}_\rho^{t+1\top}\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{u}^t}\frac{\partial \boldsymbol{u}^t}{\partial \boldsymbol{w}^t}\right)^\top \tag{21}$$

We compute $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{u}^t}$ first, then pre-multiply by $\boldsymbol{r}_\rho^{t+1\top}$ to produce a vector, and do one DST/DCT transform on that vector.

We exploit the sparsity patterns of these matrices to improve performance. $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{u}^t}$ always has d entries per row, in the same positions. We allocate one sparse matrix with this structure and reuse it. $\frac{\partial \boldsymbol{\rho}^{t+1}}{\partial \boldsymbol{\rho}^t}$ has 4 (8) entries per row in 2D (3D), but they could be at any of 9 (27) positions in 2D (3D). We allocate one sparse matrix with 9 (27) entries per row, then fill entries or set to zero as needed.

Table 1: Timing statistics and parameter values for each example

Example	Grid Resolution	# Basis	# Frames	# Keyframes	Forward Simulation (s)	Derivative Computation (s)
Blobby (Fig. 1 (a))	1024×1024	100	20	2	703.788	1192.514
Jellyfish (Fig. 1 (b))	1024×1024	100	20	2	606.923	996.961
Baby Dragon (Fig. 4)	512×512	16	20	2	19.136	25.549
Bird (Fig. 7)	128×128	100	40	1	47.939	43.149
Bunny (Fig. 8 (f))	128×128	400	5	1	5.327	44.032
GRAPH (Fig. 9)	256×256	100	40	4	147.349	193.645
F/L/U/I/D (Fig. 11)	128×128	100	40	1	45.497	41.737
Running Man (Fig. 13 (c))	128×128	100	30	15	58.791	60.439
Interactive forward pathline control (Fig. 6 (b))	256×256	100	5	0	0.757	0.580
Interactive density control (Fig. 10(d))	256×256	100	5	1	3.306	3.744

4.4 Obstacle derivatives

Our obstacle cost function ξ_o only depends on the velocity, so $\frac{\partial \xi_o}{f^t} = 0$ and $\frac{\partial \xi_o}{\rho^t} = 0$. The gradient w.r.t. velocity is $\frac{\partial \xi_o}{\partial w^t} = \frac{\partial \xi_o}{\partial u^t}\frac{\partial u^t}{\partial w^t}$, where each element of $\frac{\partial \xi_o}{\partial u^t}$ is (dropping t superscript for clarity)

$$\frac{\partial \xi_o}{u_x} = 2\left(u_x \cdot \tilde{u}_x\right)\left(u_x^\top \frac{\partial \tilde{u}_x}{\partial u_x} + \tilde{u}_x^\top\right) \tag{22}$$

As with the density derivatives, $\frac{\partial u^t}{\partial w^t}$ is a DST/DCT transform and is done once on the $\frac{\partial \xi_o}{\partial u^t}$ vector.

5 RESULTS

5.1 Implementation Details

We based our implementation on the C++ Eigenfluids implementation provided by [Cui et al. 2018]. We utilize the Eigen library [Guennebaud et al. 2010], FFTW [Frigo and Johnson 2005], and the NLopt [Johnson 2007] implementation of L-BFGS. All results were run on a Macbook Pro with an M2 Pro and 16GB of RAM. Timing and parameter values are provided in Table 1. Our scaling parameters were all set to 1.0, except for k_d, which ranged from 0.005 to 1.0. Examples with more pronounced movement or deformation tended to look better with a smaller k_d. Animation sequences are shown in our accompanying video.

5.2 Keyframe control

Our method works successfully with a variety of keyframes. In Fig. 9, we transition between the letters G-R-A-P-H over a sequence of 40 frames. At a resolution of 256×256, this optimization takes us roughly 5 minutes, significantly faster than previous methods at this resolution. In Fig. 1, we optimize fluid flows between hand-drawn artwork frames. We are able to match the keyframes in shape well; the color information does not factor into our cost function. These are our slowest examples, as they are our highest resolution 1024×1024. Still, we optimize these on the order of 10s of minutes, whereas previous methods at lower resolutions are on the order of hours. The baby dragon in Fig. 4 is very fast because it uses a small velocity basis. In Fig. 13, an initial optimization is run with 3 keyframes. However, the keyframes are far apart in time, and the inbetween motion may not look correct. Our system can quickly update the solution when new keyframes are added.

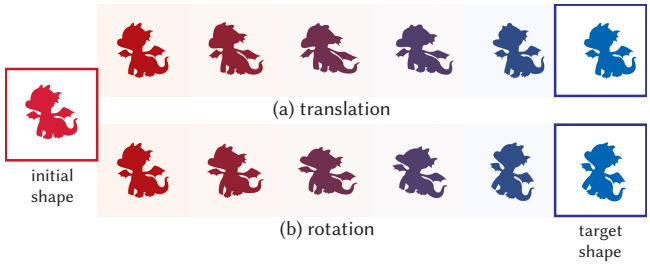

Figure 4: Eigenfluids provides a separation between velocity and density DOFs. Here, a high-resolution dragon shape (512x512) is transformed to different keyframes using only $r = 16$ basis functions.

5.3 Multi-resolution

The crown example shown in Fig. 3 cannot converge at higher resolutions. The keyframes are spatially non-overlapping, so our cost function gives zero gradients. The lower-resolution blurring of the keyframes creates overlap, allowing the optimizer to make progress. The multi-resolution sequence is also shown for Blobby and Jellyfish in Fig. 1.

5.4 Frequency control

Our frequency cascade can help the optimizer find better minima. In Fig. 12, we show the frequency cascade captures keyframe features better and reaches a lower objective value. The cascade also speeds convergence, because 1) as noted by [Tang et al. 2021], focusing on lower frequencies first can help the optimizer, and 2) by reducing the number of basis functions we reduce our DOFs, so optimization is dramatically faster as the basis size decreases.

5.5 Obstacles

Obstacles are robustly handled with our modified cost function. In Fig. 5, a plume of smoke rises and curls around a solid object. We set one target density keyframe (created from a smoke simulation

with no objects) and show the effects of changing the k_d parameter, giving artists a balance between obstacle accuracy and keyframe matching accuracy.

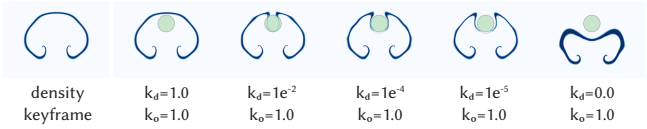

Figure 5: A smoke plume rises around an object. Parameter values control how well it matches the final keyframe.

5.6 Real-time interactive control

While some of our initial keyframe based optimizations are not real-time, we found our method fast enough to offer interactive feedback and control on subsequent edits. In Fig. 10, we show how a user can modify a density keyframe after the initial solve. Our system uses the initial solution as a warm start and can reoptimize interactively. Beyond density keyframes, users can also adjust velocity keyframes and pathlines (Fig. 6). Our system can interactively draw pathlines forward or backward, and adjust to user pathline changes in real-time.

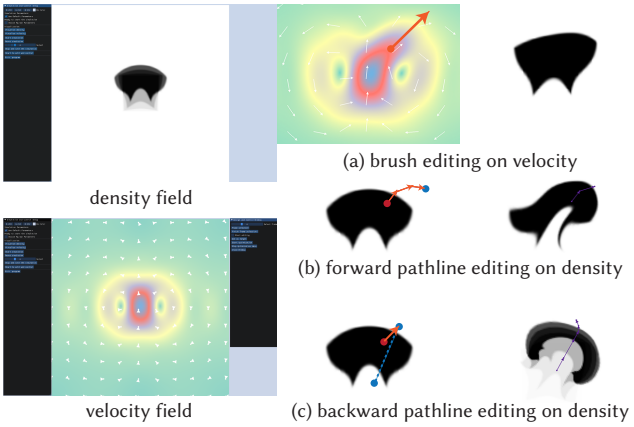

Figure 6: Starting with a simulation of smoke moving upwards (left), the user can (a) edit the velocity field. They can (b) choose a point and edit the end of its pathline in a future frame (blue dot). When clicking a point, they can also (c) trace its path backwards and specify points along the pathline.

5.7 Comparisons

We recreate several examples from previous papers to show the efficacy of our method. In the bird example from [Pan and Manocha 2017] (Fig. 7), we match the keyframe with fewer high-frequency artifacts. An interesting note here is that due to the strict incompressibility of our method, there are several small gaps which cannot be closed. This is expected, and starting from a torus (which has the same topology as the desired shape) achieves the keyframe with no gaps.

In Fig. 11, we transform the circle shape into five different letters over 40 timesteps. Compared to previous work [Pan and Manocha 2017; Tang et al. 2021], our method generates fewer artifacts during intermediate states, most likely due to the reduced velocity representation. Our method is also much faster, taking 86 seconds compared to 15 minutes or more for previous methods.

Standard semi-lagrangian advection (for density) is applied for its simplicity. We emphasize that our method produces significantly better results with this simple advection than previous methods, which use more advanced advection schemes.

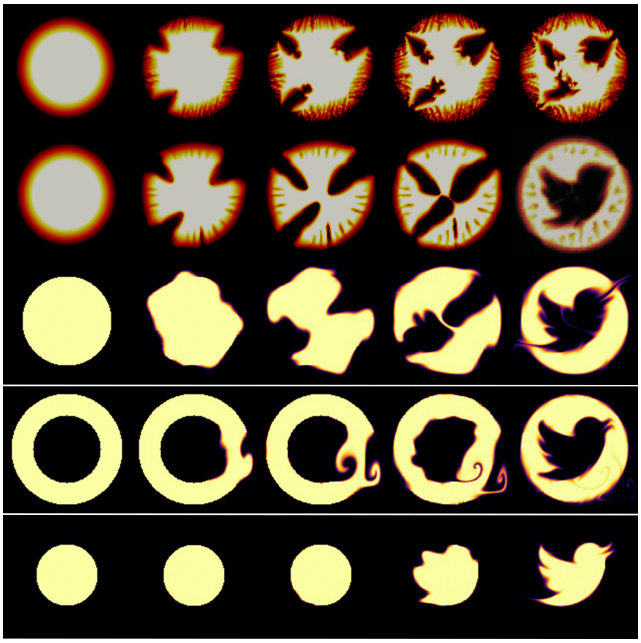

Figure 7: We transform a circle keyframe (left) into a bird shape (right). Pan and Manocha [2017] had many artifacts when using semi-lagrangian advection (top row), so they used an up-winding scheme (second row). Our method converges well when using semi-lagrangian advection for the density (bottom three rows).

6 DISCUSSIONS AND CONCLUSION

In this work, we have showcased the utilization of Eigenfluids for fluid control, highlighting its high efficiency and potential for interactive applications. Our approach also integrates pathline control with keyframes, enhancing iterative workflows. We have provided analytical expressions for the necessary gradients, enabling straightforward implementation. The results presented in this paper demonstrate the efficacy of multi-resolution and frequency control.

Nevertheless, we have only focused on 2D applications in this paper, leaving the implementation and validation of 3D fluid control untouched. Theoretically, the Eigenfluid method and our gradient expressions can be extended to 3D, but the scalability of this method is anticipated to diminish when transitioning from 2D to 3D due to the intrinsic expansion of the basis functions. This deserves more analysis and study.

Additionally, a large number of basis functions might be required to handle complex obstacles accurately or to accommodate extreme deformations between keyframes, which will reduce performance. Exploring methods for compressing the advection tensor could mitigate some of these challenges, presenting an intriguing direction for future work.

More sophisticated frequency control could be explored through manipulation of the advection tensor. This governs the flow of energy in Eigenfluids and could be used to give users more control. Exploring alternative objectives, such as partial keyframes for sparser constraints or objectives incorporating color information, could broaden the method's applicability. Moreover, most of our method would map well to GPU architectures, enabling even more efficient and large-scale fluid control.

Finally, fluid simulation and control with free surfaces between liquid and gas are still unexplored in this particular reduced space, which highlights another potential area for future development.

ACKNOWLEDGMENTS

We express our gratitude to all reviewers for their valuable feedback and suggestions on this work. We thank Qiaodong Cui, Jingwei Tang, and Zherong Pan for sharing their codes and offering help for comparisons. We would also like to thank John Hancock and Xuan Dam for their essential administrative support. Special thanks to Masha Shugrina for drawing the beautiful teaser keyframes. We appreciate the support from Yiting Li and Zhecheng Wang, who assisted with video editing and proofreading. This work is supported by funding from the NSERC Discovery Grant, the Ontario Early Researchers Award, the Canada Research Chairs Program, and gifts from Adobe Research and Autodesk.

REFERENCES

Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *Symposium on Computer Animation (SCA 06)*. ACM, pages–25.

Joshua Aurand, Raphael Ortiz, Silvia Nauer, and Vinicius C Azevedo. 2022. Efficient Neural Style Transfer for Volumetric Simulations. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–10.

Robert Bridson. 2015. *Fluid Simulation for Computer Graphics* (2nd ed.). A K Peters/CRC Press, New York.

Mengyu Chu, Nils Thuerey, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. 2021. Learning meaningful controls for fluids. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable laplacian eigenfluids. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Tyler De Witt, Christian Lessig, and Eugene Fiume. 2012. Fluid simulation using laplacian eigenfunctions. *ACM Transactions on Graphics (TOG)* 31, 1 (2012), 1–11.

M-L Eckert, Wolfgang Heidrich, and Nils Thuerey. 2018. Coupled fluid density and motion from single views. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 47–58.

Raanan Fattal and Dani Lischinski. 2004. Target-driven smoke animation. *ACM Trans. Graph.* 23, 3 (aug 2004), 441–448. https://doi.org/10.1145/1015706.1015743

Sean Flynn, Parris Egbert, Seth Holladay, and Bryan Morse. 2019. Fluid carving: intelligent resizing for fluid simulation data. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Nick Foster and Dimitris Metaxas. 1997. Controlling fluid animation. In *Proceedings computer graphics international*. IEEE, 178–188.

Matteo Frigo and Steven G. Johnson. 2005. The Design and Implementation of FFTW3. *Proc. IEEE* 93, 2 (2005), 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation".

James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. 2014. From capture to simulation: connecting forward and inverse problems in fluids. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Philipp Holl, Vladlen Koltun, and Nils Thuerey. 2020. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457* (2020).

Jeong-mo Hong and Chang-hun Kim. 2004. Controlling fluid animation with geometric potential. *Computer Animation and Virtual Worlds* 15, 3-4 (2004), 147–157.

Ruoguan Huang, Zeki Melek, and John Keyser. 2011. Preview-based sampling for controlling gaseous simulations. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 177–186.

Tiffany Inglis, M-L Eckert, James Gregson, and Nils Thuerey. 2017. Primal-dual optimization for fluids. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 354–368.

Steven G. Johnson. 2007. The NLopt nonlinear-optimization package. https://github.com/stevengj/nlopt.

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2019. Transport-based neural style transfer for smoke simulations. *ACM Trans. Graph.* 38, 6, Article 188 (nov 2019), 11 pages. https://doi.org/10.1145/3355089.3356560

Byungsoo Kim, Xingchang Huang, Laura Wuelfroth, Jingwei Tang, Guillaume Cordonnier, Markus Gross, and Barbara Solenthaler. 2022. Deep Reconstruction of 3D Smoke Densities from Artist Sketches. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 97–110.

Yootai Kim, Raghu Machiraju, and David Thompson. 2006. Path-based control of smoke simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 33–42.

Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-reduced variational fluid simulation. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–12.

Jamie Madill and David Mould. 2013. Target particle control of smoke simulation. In *Proceedings of Graphics Interface 2013*. 125–132.

Pierre-Luc Manteaux, Ulysse Vimont, Chris Wojtan, Damien Rohmer, and Marie-Paule Cani. 2016. Space-time sculpting of liquid animation. In *Proceedings of the 9th International Conference on Motion in Games*. 61–71.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)* 23, 3 (2004), 449–456.

Viorel Mihalef, Dimitris Metaxas, and Mark Sussman. 2004. Animation and control of breaking waves. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 315–324.

M. Muller, M. Teschner, and M. Gross. 2004. Physically-based simulation of objects represented by surface meshes. In *Proceedings Computer Graphics International, 2004.* 26–33.

Michael B Nielsen and Robert Bridson. 2011. Guide shapes for high resolution naturalistic liquid simulation. In *ACM SIGGRAPH 2011 papers*. 1–8.

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.

Michael B Nielsen, Brian B Christensen, Nafees Bin Zafar, Doug Roble, and Ken Museth. 2009. Guiding of smoke animations through variational coupling of simulations at different resolutions. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 217–226.

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), 1–10.

Zherong Pan and Dinesh Manocha. 2017. Efficient solver for spacetime control of smoke. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1.

Frédéric Pighin, Jonathan M Cohen, and Maurya Shah. 2004. Modeling and editing flows using advected radial basis functions. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 223–232.

Nick Rasmussen, Doug Enright, Duc Nguyen, Sebastian Marino, Nigel Sumner, Willi Geiger, Samir Hoon, and Ron Fedkiw. 2004. Directable Photorealistic Liquids. In *Symposium on Computer Animation*, R. Boulic and D. K. Pai (Eds.). The Eurographics Association. https://doi.org/10.2312/SCA/SCA04/193-202

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

Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. 2014. Blending liquids. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Bo Ren, Chen-Feng Li, Ming C Lin, Theodore Kim, and Shi-Min Hu. 2013. Flow field modulation. *IEEE Transactions on Visualization and Computer Graphics* 19, 10 (2013), 1708–1719.

Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. 2018b. Example-based turbulence style transfer. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–9.

Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita. 2018a. Editing fluid animation using flow interpolation. *ACM Transactions on Graphics (TOG)* 37, 5 (2018), 1–12.

Connor Schenck and Dieter Fox. 2018. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*. PMLR, 317–335.

Arnaud Schoentgen, Pierre Poulin, Emmanuelle Darles, and Philippe Meseure. 2020. Particle-based Liquid Control using Animation Templates. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 79–88.

Joshua Schpok, William Dwyer, and David S. Ebert. 2005. Modeling and Animating Gases with Simulation Features. In *Symposium on Computer Animation*, D. Terzopoulos, V. Zordan, K. Anjyo, and P. Faloutsos (Eds.). The Eurographics Association. https://doi.org/10.2312/SCA/SCA05/097-106

Lin Shi and Yizhou Yu. 2005a. Controllable smoke animation with guiding objects. *ACM Transactions on Graphics (TOG)* 24, 1 (2005), 140–164.

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

Tetsuya Takahashi and Ming C Lin. 2019. Video-guided real-to-virtual parameter transfer for viscous fluids. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12.

Jingwei Tang, Vinicius C. Azevedo, Guillaume Cordonnier, and Barbara Solenthaler. 2021. Honey, I Shrunk the Domain: Frequency-aware Force Field Reduction for Efficient Fluids Optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 339–353.

Jingwei Tang, Byungsoo Kim, Vinicius C Azevedo, and Barbara Solenthaler. 2023. Physics-Informed Neural Corrector for Deformation-based Fluid Control. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, 161–173.

Nils Thuerey. 2016. Interpolations of smoke and liquid simulations. *ACM Transactions on Graphics (TOG)* 36, 1 (2016), 1–16.

Nils Thürey, Richard Keiser, Mark Pauly, and Ulrich Rüde. 2009. Detail-preserving fluid control. *Graphical Models* 71, 6 (2009), 221–228.

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe control of smoke simulations. In *ACM SIGGRAPH 2003 Papers*. 716–723.

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based smoke with vortex shedding and variational reconnection. In *ACM SIGGRAPH 2010 papers*. 1–12.

Haoran Xie, Keisuke Arihara, Syuhei Sato, and Kazunori Miyata. 2022. Dualsmoke: Sketch-based smoke illustration design with two-stage generative model. *arXiv preprint arXiv:2208.10906* (2022).

Guowei Yan, Zhili Chen, Jimei Yang, and Huamin Wang. 2020. Interactive liquid splash modeling by user sketches. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–13.

Ben Yang, Youquan Liu, Lihua You, and Xiaogang Jin. 2013. A unified smoke control method based on signed distance field. *Computers & graphics* 37, 7 (2013), 775–786.

Zhi Yuan, Fan Chen, and Ye Zhao. 2011. Pattern-guided smoke animation with lagrangian coherent structure. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–8.

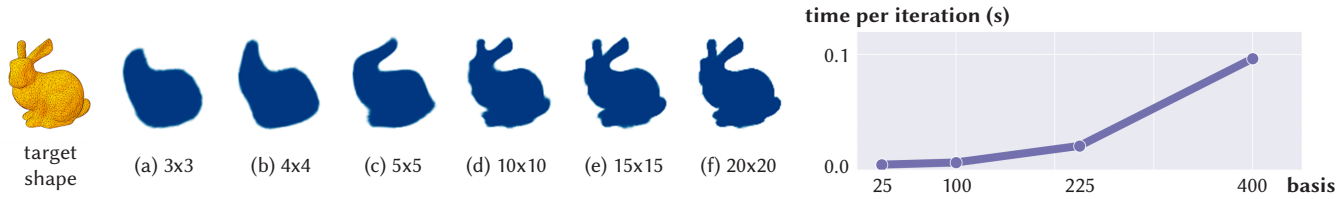

Figure 8: Starting from a circle, we morph to a bunny keyframe using different amounts of basis functions, which controls the complexity of deformations we can model. We found that 100 basis functions (d) gave good results for our examples and maintained good efficiency.

Figure 9: With smoke initialized to the letter G, our system finds a flow to transition to the letters R A P H.

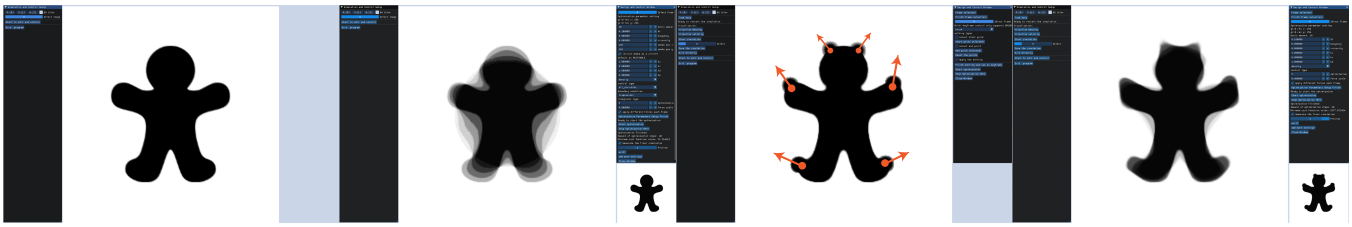

Figure 10: After initially warping a gingerbread man to a bear, the user interactively modify the bear keyframe.

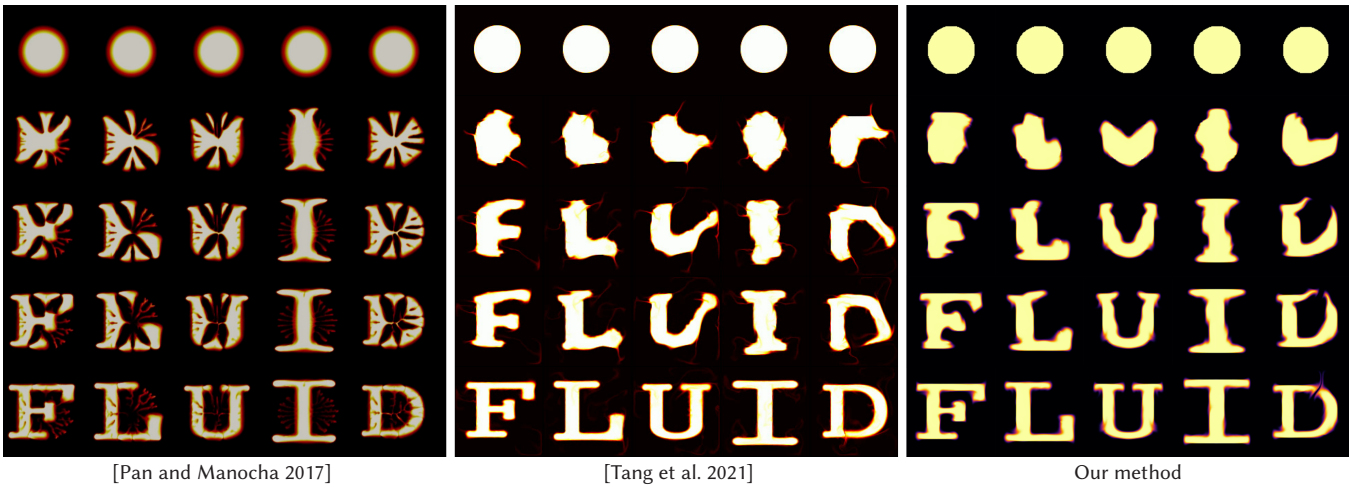

Figure 11: We morph a circle keyframe into different letters and compare the results of two recent methods. The resolution is 128^2 and from top to bottom, we show the results at frames 0, 20, 30, 35, and 40.

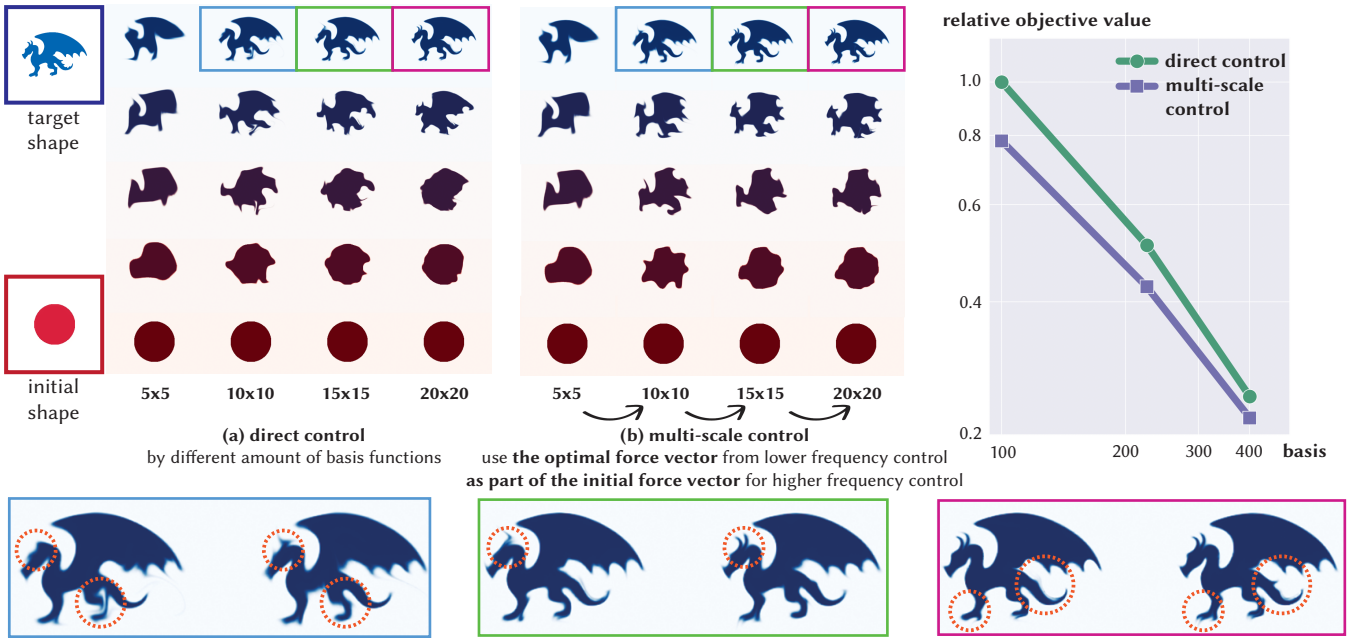

Figure 12: Our frequency cascade allows the optimizer to match keyframes more accurately, highlighted in the bottom row, and shown quantitatively as lower objective values.

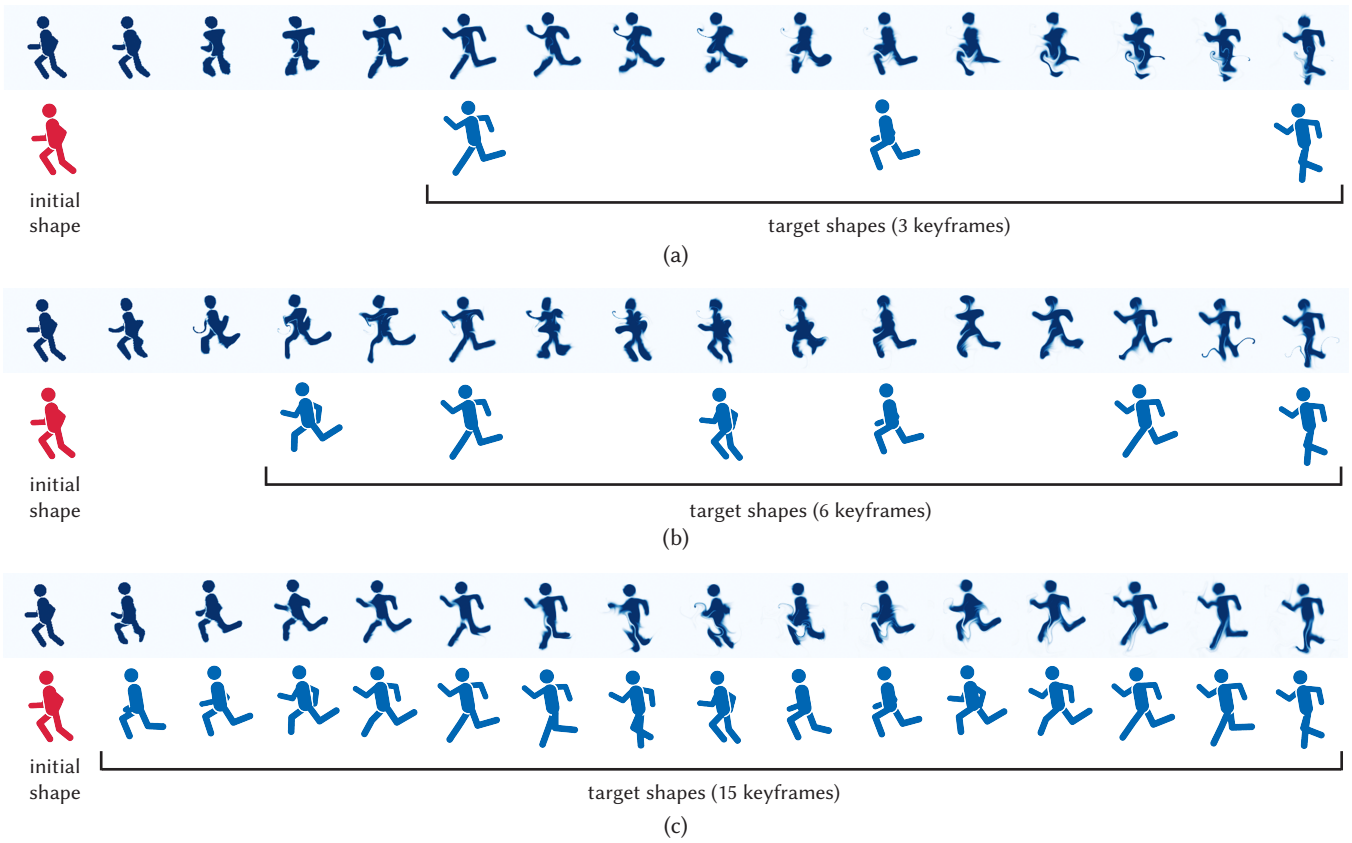

Figure 13: When keyframes are too far apart (a), the generated motion may not capture the users intent. Users can quickly add new keyframes, (b) and (c), until the motion matches what they expect.