Editing Fluid Animation using Flow Interpolation

SYUHEI SATO, DWANGO Co., Ltd., Dwango CG Research
YOSHINORI DOBASHI, Hokkaido University and Dwango CG Research
TOMOYUKI NISHITA, Dwango CG Research and Hiroshima Shudo University

Fig. 1. A large-scale sandstorm animation (right) created by combining two input flows simulated in a small-scale area (left).

The computational cost for creating realistic fluid animations by numerical simulation is generally expensive. In digital production environments, existing precomputed fluid animations are often reused for different scenes in order to reduce the cost of creating scenes containing fluids. However, applying the same animation to different scenes often produces unacceptable results, so the animation needs to be edited. In order to help animators with the editing process, we develop a novel method for synthesizing the desired fluid animations by combining existing flow data. Our system allows the user to place flows at desired positions, and combine them. We do this by interpolating velocities at the boundaries between the flows. The interpolation is formulated as a minimization problem of an energy function, which is designed to take into account the inviscid, incompressible Navier-Stokes equations. Our method focuses on smoke simulations defined on a uniform grid. We demonstrate the potential of our method by showing a set of examples, including a large-scale sandstorm created from a few flow data simulated in a small-scale space.

CCS Concepts: • **Computing methodologies** → **Animation**; *Physical simulation*;

Additional Key Words and Phrases: fluid simulation, reusing fluid animations, inviscid incompressible Navier-Stokes equations, interpolating velocity fields

Authors' addresses: Syuhei Sato, DWANGO Co., Ltd., Dwango CG Research, syuhei_sato@dwango.co.jp; Yoshinori Dobashi, Hokkaido University, Dwango CG Research, doba@ime.ist.hokudai.ac.jp; Tomoyuki Nishita, Dwango CG Research, Hiroshima Shudo University, nishita@shudo-u.ac.jp.

0730-0301/2018/0-ART0 $15.00
https://doi.org/0000001.0000001_2

ACM Reference Format:
Syuhei Sato, Yoshinori Dobashi, and Tomoyuki Nishita. 2018. Editing Fluid Animation using Flow Interpolation. *ACM Trans. Graph.* 0, 0, Article 0 (2018), 12 pages. https://doi.org/0000001.0000001_2

1 INTRODUCTION

Physically-based simulation of fluids has become an important element in many computer graphics applications, such as movies and computer games. However, one problem is the expensive computational cost, which makes it difficult to create the desired fluid animation since the simulation must be repeated multiple times to find an appropriate set of parameters that can produce a satisfactory result.

In digital production environments, existing precomputed fluid animations are often reused for different scenes. Although this approach reduces the computational cost significantly, applying the same animations to different scenes often produces unacceptable results. Therefore, the animation usually needs to be edited. Several methods have therefore been proposed for synthesizing new fluid animations by reusing precomputed flow data [Raveendran et al. 2014; Sato et al. 2016, 2015; Thuerey 2016]. These methods allow us to create plausible fluid animations different from input animations. However, since the purpose of these methods is to perform simple interpolation or deformation, the physical properties of the fluid are not considered. Our goal is to obtain physically more plausible edited results than these methods by considering the physical properties.

Cut-and-paste operations are common in editing images, videos, and even documents. Although there are many methods for seamlessly combining multiple images/videos, no methods have been proposed for fluid flow. We cannot simply apply the methods used for images/videos to flows because they do not pay any attention to the physical properties of the fluid. Our method addresses this problem and we present a novel mathematical framework that allows

cut-and-paste editing of multiple flows in a way that respects the important physical properties. In our system, the user can crop the desired regions from one flow and paste it onto another, and these are seamlessly combined by interpolating the flow at the boundaries of the cropped regions. We formulate the interpolation as a minimization problem of our energy function consisting of two terms: the difference of the curl of the flow and the divergence of the flow. These terms are designed to consider the inviscid, incompressible Navier-Stokes equations. Our energy function is derived from the Dirichlet energy to generate smooth and plausible velocity fields.

We demonstrate the potential of our method by showing several examples. We can create different fluid animations from a single input flow. We can also create very large-scale flow by repeatedly combining a few sets of flow data. Fig. 1 shows such an example where animation of a very large sandstorm is created.

2 RELATED WORK

Physically-based fluid simulation. Fluid simulations based on the Navier-Stokes equations were firstly introduced to computer graphics by Foster and Metaxas [1996], and later Stam [1999] developed an unconditionally stable solver. Since these methods were introduced, many methods have been proposed for simulating various fluid phenomena, such as smoke [Fedkiw et al. 2001], water [Foster and Fedkiw 2001], fire [Nguyen et al. 2002], explosion [Feldman et al. 2003; Yngve et al. 2000], and cloud [Miyazaki et al. 2002]. Bridson published a great textbook that summarized various simulation methods [Bridson 2015]. Although these methods can produce realistic fluid animation, the user must repeatedly execute fluid simulations with different parameter settings until the desired animation is created.

Control methods. Some control methods have been proposed for creating fluid animations with desired shapes. Treuille et al. [2003] optimized the control forces such that smoke simulation is matched to target density and velocity fields at specified keyframes. Fattal et al. [2004] controlled smoke simulation by introducing additional external forces calculated based on target density distributions. McNamara et al. [2004] developed a method for optimizing control forces using the adjoint method. Thürey et al. [2006] controlled liquid simulation while preserving the small-scale details by applying control forces to the coarse-scale components of the velocity field. Nielsen et al. [2010; 2009] controlled a high resolution simulation based on a low resolution preview simulation by solving a minimization problem. Huang et al. [2011] proposed a method for guiding the high resolution simulation by locally adjusting the velocity and density fields according to the low resolution preview simulation. Bhattacharya et al. [2012] used the steady state Stokes flow for controlling liquid simulations while capturing the rotational components. Pan et al. [2013] interactively edited a low-resolution fluid simulation and guided a high-resolution simulation according to the edited simulation. However, with these control methods, the full simulation must be executed and some control/physical parameters must be adjusted appropriately. Therefore, the full simulation with different parameter settings must be repeated in order to create the desired animation. Furthermore, these methods are not suitable for our purpose since they are not designed to combine multiple precomputed flows.

Re-simulation methods. Kim and Delaney [2013] presented an efficient re-simulation method. This method generates basis functions by applying principal component analysis (PCA) to a single set of velocity fields, and then fluids with different parameter settings can be efficiently re-simulated in the basis space. In this method, the results are limited to those represented by a linear combination of the basis functions. Furthermore, the memory consumption in applying PCA is a bottleneck for a large database. Bojsen-Hansen and Wojtan [2016] also proposed a fluid re-simulation method by introducing non-reflecting boundary conditions with inflow/outflow constraints. This method allows the user to locally re-simulate the fluid as a post-process. However, this method cannot use the semi-Lagrangian method and FLIP for the advection term, so the time step must be small. Therefore, the computational cost is relatively high. By contrast, our method uses the semi-Lagrangian method for the advection term and computes flows only at the boundaries between the input flows. Therefore, our system also allows to create large-space flow at low cost, not only the local editing and, again, these methods are not suitable for combining multiple precomputed flows.

Reusing existing fluid animations. Several methods have been proposed for efficiently creating new fluid animations by reusing existing flow data. Raveendran et al. [2014] developed a method for smoothly blending existing free surface liquid animations. This method semi-automatically matches 3D triangular meshes from existing liquid simulations to plausibly interpolate between the input animations. Thuerey [2016] proposed an interpolation method for smoke and liquid represented by grid-based signed-distance functions. This method computes deformations between the input signed-distance functions by using optical flow. Sato et al. [2015] showed 3D velocity fields could be deformed while guaranteeing the incompressibility of the flow using vector potentials. However, drastic changes in flows are difficult to achieve using these deformation-based methods. By contrast, our method achieves relatively large changes by partially combining multiple flows. To combine existing fluid animations, Sato et al. [2016] proposed an interpolation method that respects incompressibility. This method generates new fluid animations by interpolating the flow at the boundaries between the input flows. However, the momentum equation in the Navier-Stokes equations is not taken into account in this method. By contrast, our method can take into account the momentum equation as well as the incompressibility in interpolating the flow. Furthermore, we can smoothly combine multiple flows temporally by finding the best-matching frames between the inputs.

Image and Video Synthesis. Our method seamlessly combines multiple flows by minimizing an energy function. Similar ideas have been proposed for seamlessly editing and combining multiple images. One of the most popular methods is the technique called Poisson image editing [Perez et al. 2003]. In this method, multiple images are seamlessly combined so that the gradients of the image intensities are preserved. This is achieved by minimizing the difference of the gradient of pixel intensities. A graph-based approach is

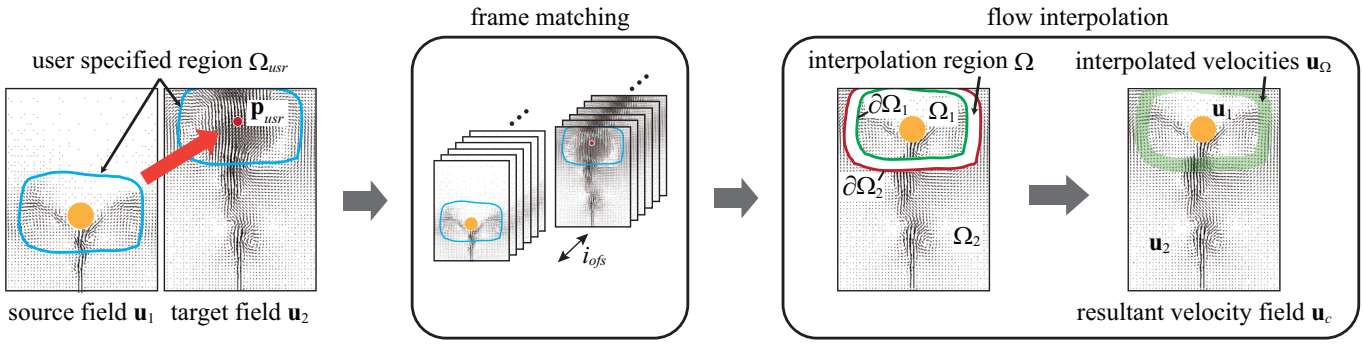

Fig. 2. Overview of our method. $\mathbf{p}_{usr}$ is a user specified position where Ω_{usr} is placed in a target field, i_{ofs} indicates a temporal offset, and $\partial\Omega_1$ and $\partial\Omega_2$ are boundaries of Ω. For visual clarity, 3D velocity fields are shown as 2D velocity fields. The orange circles indicate obstacles.

also frequently used to seamlessly combine multiple images. Kwatra et al. [2003] developed a method for synthesizing the texture from multiple images using graph cuts. We can apply these techniques to velocity fields of the fluid but they do not take into account the physical properties. Our method can be considered as an extension of this technique to the velocity fields so that the important physical properties are incorporated.

Geometry Processing. Determining vector fields by prescribing the curl and divergence have previously been used in geometry processing. Wang et al. [2006] introduced the prescribing curl and divergence in designing vector fields, and Fisher et al. [2007] extended Wang's method. Fisher's method satisfies a sparse set of user-provided constraints (sources, sinks and vortices and/or arbitrary vectors) over arbitrary triangle meshes. Then, by minimizing the error between unknown vector fields and these constraints, the smooth and desired vector fields can be obtained. Similarly, our method interpolates flow by minimizing the error of the curl and the magnitude of the divergence. However, our method is designed to smoothly combine multiple dynamic 3D flows obtained by fluid simulation.

3 OUR COMBINATION METHOD

3.1 Overview

Fig. 2 shows an overview of our method. First, the user prepares the input velocity field sequences $\mathbf{u}$ obtained using a fluid simulation. For the sake of clarity, we explain our method for the case in which there are two input velocity fields, and these are denoted as $\mathbf{u}_1$ and $\mathbf{u}_2$ as shown in Fig. 2. We use these two velocity fields throughout the paper unless otherwise stated. Let us call the two fields, $\mathbf{u}_1$ and $\mathbf{u}_2$, the source and target fields, respectively. The user specifies an arbitrary region Ω_{usr} (the region enclosed by the blue curve in Fig. 2) in the source field, and specifies a position $\mathbf{p}_{usr}$ in the target field (the red point in Fig. 2), where the source field cropped by Ω_{usr} is copied. Simultaneously, the same region as Ω_{usr} is defined in the target field so that the center of Ω_{usr} is at $\mathbf{p}_{usr}$ (Fig. 2). Then, the two velocity fields are combined.

The method consists of two processes: frame matching and flow interpolation. In the frame matching process, we found the frames where the distributions between two inputs are the most similar in Ω_{usr}. A correlation function between the two inputs is computed based on the input density fields, and then we compute a temporal offset i_{ofs} such that the correlation is maximized. This frame matching process is executed only once as a preprocess and does not need to be repeated as long as Ω_{usr} or $\mathbf{p}_{usr}$ are unchanged. Note that our system allows the user to manually specify i_{ofs}, instead of using our automatic frame matching method if the user is not satisfied with the automatic result.

Next, in the interpolation process, the flow in Ω_{usr} in the source field is copied into the target field. An interpolation region, Ω, with a predefined width is automatically created around the boundary of Ω_{usr} as shown in Fig. 2. The width of the interpolation region is specified by the user (we experimentally found that an appropriate width is about 2 – 7% of the entire simulation space). We denote the inner and outer regions of Ω_{usr} as Ω_1 and Ω_2 respectively, except Ω (Fig. 2). The boundaries between Ω and Ω_1, Ω and Ω_2 are denoted by $\partial\Omega_1$, $\partial\Omega_2$ (green and red curves in Fig. 2), respectively. Then, our system interpolates the velocities $\mathbf{u}_\Omega$ in Ω by minimizing an energy function in a way that respects the important physical properties of the fluids: the momentum equation and the incompressibility. We develop the energy function through an L2 norm of the gradients of the velocities (the definition of the energy function is given in Sec. 3.4). Finally, we obtain the resultant velocity field $\mathbf{u}_c$: $\mathbf{u}_c = \mathbf{u}_\Omega$ at Ω, $\mathbf{u}_c = \mathbf{u}_1$ at Ω_1, and $\mathbf{u}_c = \mathbf{u}_2$ at Ω_2 (Fig. 2). To visualize the flow, smoke densities are advected by the resultant velocity field $\mathbf{u}_c$.

3.2 Input Velocity and Density Fields

One of the inputs to our system is a set of incompressible velocity fields $\mathbf{u}_k(i)$, where $i = 1, 2, \cdots, I$ represents the frame number, and I is the number of frames in the input velocity field, k is an indicator; $k = 1$ and $k = 2$ indicate source and target fields, respectively. These velocity fields are obtained by numerically solving the following inviscid, incompressible Navier-Stokes equations (or incompressible Euler equations).

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

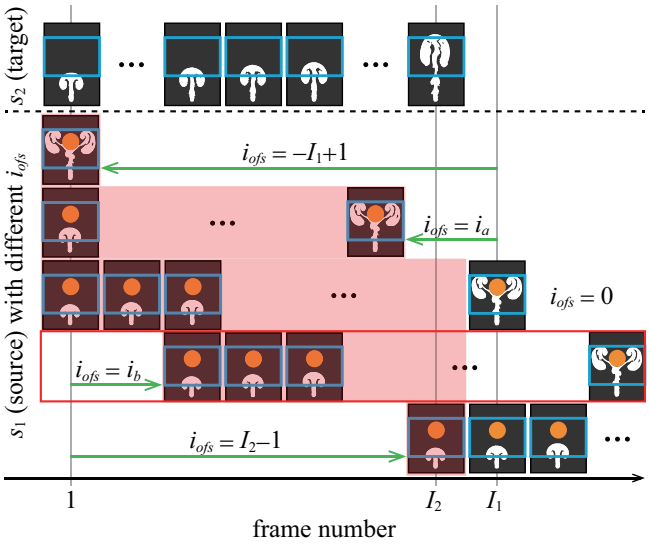

Fig. 3. Matching frames between the source field and the target field. This figure shows the five cases with different i_{ofs} ($-I_1+1$, i_a, 0, i_b, I_2-1), and for $i_{ofs} = 0$ the full length I_1 is shown. White and black colors indicate $s = 1$ (fluid) and $s = -1$ (not fluid), respectively. i_{ofs} is a temporal offset (shown by green arrows). The blue rectangles indicate the user specified region Ω_{usr}, the red shaded regions indicate the range for computing a correlation function. In the example of this figure, the correlation function is maximized for $i_{ofs} = i_b$ (denoted as a red rectangle).

where ρ is the fluid density, p is the fluid pressure, and $\mathbf{f}$ represents the external forces. Eq. (1) is the momentum equation, and Eq. (2) is the incompressibility condition. For smoke simulation, we compute Eqs. (1) and (2) based on the stable fluid method developed by Stam [1999]. The smoke densities $D_k(i)$ are passively advected in line with $\mathbf{u}_k(i)$. The sequences of the density field are also the inputs to our system.

3.3 Frame Matching

We first find the best-matching frames so that the similarity between the input flows in the user specified region Ω_{usr} is maximized. In addition, we want to maximize the number of temporally overlapping frames in order to preferably increase the number of frames for resulting animations. To achieve this, we compute a correlation function between the input density fields and calculate a temporal offset such that the correlation is maximized. In the following, we first describe the frame matching algorithm for the case of two input flows. The method for the case of three or more input flows is explained later.

First, our method creates sequences of grid data $s_k(i)$ storing a Boolean value representing the presence of smoke at each grid point, computed from the density fields $D_k(i)$. $s_k(i)$ is set to 1 for the grid points satisfying $D_k(i) > D_{th}$, and set to -1 otherwise. D_{th} is a threshold value for the density which is chosen so that the smoke region is extracted.

We then compute the correlation between s_1 and s_2 with different temporal offsets and find the optimal offset i_{ofs} by solving the following maximization problem.

$$\arg\max_{i_{ofs}} \sum_{i=I_s}^{I_e} \sum_{\Omega_{usr}} s_1(i - i_{ofs}) s_2(i). \tag{3}$$

The correlation between s_1 and s_2 is computed by using only the frames that are temporally overlapping (see the red shaded frames in Fig. 3). That is, the range of i_{ofs} is $[-I_1+1, I_2-1]$, I_s is $\max(1, i_{ofs}+1)$, I_e is $\min(I_1 + i_{ofs}, I_2)$ where I_1 and I_2 are the number of frames for the source and target fields, respectively. Note that I_1 and I_2 can be different. The final animation starts at $i = 1$ and ends at $i = I_e$. The source fields that do not temporally overlap with target fields are not used. We try all possible offsets to obtain the optimal i_{ofs}.

When n input flows are combined, the user specifies one of them as the target field and the rest of them are treated as source fields. Then, for each source field, the optimal offset frame is calculated by maximizing the correlation function between each source density field and the target density field. The number of frames for the final animation is equal to the minimum of I_e for all the source fields.

The effectiveness of this process is shown in Sec. 4 (see Figs. 6 (c) and (f)). If the user wants to combine the flows in a specific frame, he/she can directly specify the offset frame. However, unless the user carefully chooses the offset frame, visual artifacts might arise as shown in Sec. 4.

Algorithm 1 Single step of flow interpolation

$\hat{\mathbf{u}}(i) \leftarrow$ AdvectVelocityField($\mathbf{u}_c(i-1)$)
$\mathbf{u}_1(i) \leftarrow$ LoadSourceVelocityField
$\mathbf{u}_2(i) \leftarrow$ LoadTargetVelocityField
$\mathbf{u}_c(i) \leftarrow$ InterpolateFlow($\mathbf{u}_1(i), \mathbf{u}_2(i), \hat{\mathbf{u}}(i)$)

3.4 Flow Interpolation

After determining the matching frames, we interpolate the velocity field by solving a minimization problem. The velocities at Ω_{usr} in the source field are copied to the target field. Then, an interpolation region Ω with a predefined width is created around the boundary of Ω_{usr} (see Fig. 2). The velocities $\mathbf{u}_\Omega(i)$ in Ω are computed such that our energy function is minimized. In the following, we define our energy function.

To obtain a smooth velocity field $\mathbf{u}_\Omega(i)$, we want to minimize the spatial change in the velocities. This can be achieved by minimizing the following L2 norm of the gradient of the velocity [Perez et al. 2003].

$$E_{tmp}(\mathbf{u}_\Omega(i)) = \sum_{\Omega} ||\nabla \mathbf{u}_\Omega(i)||^2, \tag{4}$$

where ∇ is the gradient operator and we set the boundary conditions $\mathbf{u}_\Omega(i) = \mathbf{u}_1(i - i_{ofs})$ at $\partial\Omega_1$ and $\mathbf{u}_\Omega(i) = \mathbf{u}_2(i)$ at $\partial\Omega_2$. Eq. (4) can be considered the Dirichlet energy of the velocity. The derivative of $E_{tmp}(\mathbf{u}_\Omega(i))$ with respect to $\mathbf{u}_\Omega(i)$ should satisfy the following equation in order to minimize $E_{tmp}(\mathbf{u}_\Omega(i))$.

$$\nabla^2 \mathbf{u}_\Omega(i) = \mathbf{0}, \tag{5}$$

where ∇^2 is the Laplace operator and $\mathbf{0}$ is a zero vector. Since the left hand side of Eq. (5) is a vector Laplacian operator, we can decompose it into the following two components by applying a standard

identity:

$$-\nabla\times(\nabla\times\mathbf{u}_\Omega(i))+\nabla(\nabla\cdot\mathbf{u}_\Omega(i))=\mathbf{0}, \tag{6}$$

where $\nabla\times$ and $\nabla\cdot$ are the curl and divergence operators, respectively. This equation is equal to the derivative of the following energy function (see Appendix for the derivation):

$$E_{tmp}(\mathbf{u}_\Omega(i))=\sum_\Omega ||\nabla\times\mathbf{u}_\Omega(i)||^2+(\nabla\cdot\mathbf{u}_\Omega(i))^2. \tag{7}$$

Our energy function is derived from the above equation. This energy function has a nice property in that it contains the divergence of the velocity field as shown in the second term in Eq. (7). Thus, by minimizing the above energy function, the divergence of $\mathbf{u}_\Omega(i)$ is also minimized. In other words, using this energy function, we can automatically take into account the incompressibility. This energy function, however, does not deal with the other important physical property, i.e., the momentum equation. In addition, minimizing this energy produces too smooth flows because the magnitude of the curl is minimized.

We extend the energy function to take the momentum equation into account and avoid too smooth flows, by introducing a velocity field $\hat{\mathbf{u}}(i)$ which is obtained by advecting our result in the previous frame. The first term of E_{tmp} is then modified so that the difference between $\mathbf{u}_\Omega(i)$ and $\hat{\mathbf{u}}(i)$ is minimized. That is, our energy function is expressed by the following equation.

$$E_\Omega(\mathbf{u}_\Omega(i))=\sum_\Omega ||\nabla\times\mathbf{u}_\Omega(i)-\nabla\times\hat{\mathbf{u}}(i)||^2+(\nabla\cdot\mathbf{u}_\Omega(i))^2. \tag{8}$$

The curl of $\hat{\mathbf{u}}(i)$ is obtained from the vorticity formulation of the Navier-Stokes equations (the curl of Eq. (1)); $\nabla\times\hat{\mathbf{u}}(i)=\nabla\times\mathbf{u}_c(i-1)-\Delta t\nabla\times\{(\mathbf{u}_c(i-1)\cdot\nabla)\mathbf{u}_c(i-1)\}$, where $\mathbf{u}_c(i-1)$ is the resultant velocity field for the $(i-1)$-th frame, and Δt is the time step. Note that, for $i=1$, we assume $\hat{\mathbf{u}}(1)=\mathbf{0}$. Therefore, our energy function for the 1-st frame is $E_\Omega(\mathbf{u}_\Omega(1))=\sum_\Omega ||\nabla\times\mathbf{u}_\Omega(1)||^2+(\nabla\cdot\mathbf{u}_\Omega(1))^2$. This is equivalent to minimizing Eq. (4).

Finally, to obtain $\mathbf{u}_\Omega(i)$, we minimize $E_\Omega(\mathbf{u}_\Omega(i))$ (Eq. (8)). By taking the derivative of Eq. (8) with respect to $\mathbf{u}_\Omega$ so that $E_\Omega(\mathbf{u}_\Omega(i))$ is minimized, we obtain:

$$-\nabla\times(\nabla\times\mathbf{u}_\Omega(i))+\nabla(\nabla\cdot\mathbf{u}_\Omega(i))=-\nabla\times(\nabla\times\hat{\mathbf{u}}(i)). \tag{9}$$

We discretize the above equation using the finite difference approximation, and use the conjugate gradient (CG) method to solve it numerically. Since the left hand side of Eq. (9) is converted back to the usual vector Laplacian form using the identity applied between Eqs. (5) and (6), the above equation is solved individually for the velocity components. The steps of this flow interpolation process are shown in Algorithm 1.

We extended the Dirichlet energy of the velocity (Eq. (4)) to incorporate the momentum equation as shown in Eq. (8). With our new energy function, when the magnitude of the curl of $\hat{\mathbf{u}}(i)$ is very large, any theoretical claims related to the smoothness or other properties of the Dirichlet energy might be discarded. However, the following experiments demonstrate that our extended energy function can produce even more plausible flows than those obtained using the Dirichlet energy (Eq. (4)). This is because our energy function takes into account the incompressibility and the momentum equation.

Fig. 4 shows the temporal change in the absolute value of the divergence of two combined results. The red curves indicate the results obtained by only copying the source field to the target field while the blue curves indicate the results obtained using our method. The vertical axes indicate the maximum value of the divergence (upper image) and the spatial average of the divergence (lower one), and the horizontal axis is the frame number. The vertical axes are normalized so that the maximum value of the red curve is one in each graph. In the red curve, the divergence is a long way from zero. This is because the flow obtained just by copying the inputs has discontinuities at the boundaries of the interpolation region. By contrast, our result (shown by the blue curve) takes into account the incompressibility and hence the divergence is closer to zero. Especially, the maximum divergence is much smaller than that obtained by copying the input flow only. However, we still have small divergence even in our method and this leads to some artifacts that could be observed. Nonetheless, our method can produce smooth and plausible flow overall, as demonstrated by several examples in Sec. 4.

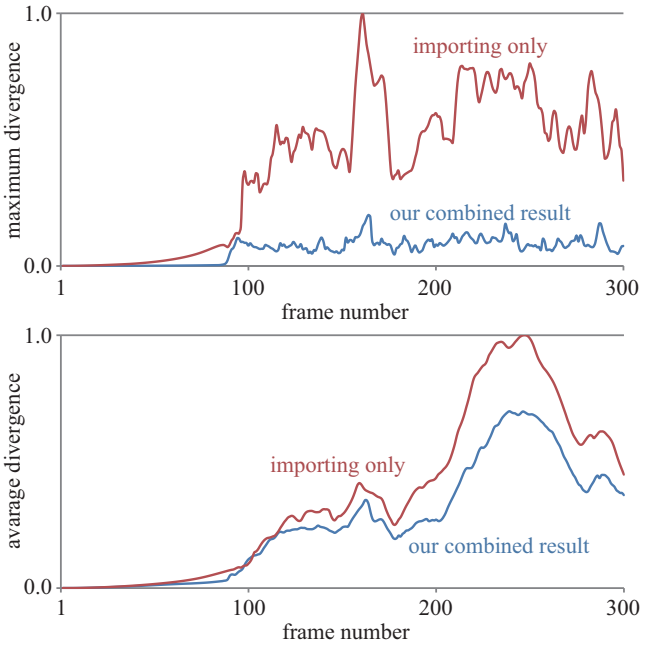

Fig. 4. A temporal change in the divergence for combined velocity fields. The maximum divergence indicates the maximum value of $|\nabla\cdot\mathbf{u}_c(i)|$ for over the simulation space. The average divergence is computed $\sum_{\Omega_{sim}}|\nabla\cdot\mathbf{u}_c(i)|/N_{sim}$, where Ω_{sim} is the entire simulation space and N_{sim} indicates the number of grid points for Ω_{sim}. The red curve indicates the transition of the divergence for a result generated by importing the velocity fields from the source field to the target field. The blue curve indicates it for our combined result.

Fig. 5 shows the temporal changes in the energy of the curl and divergence terms for two synthesized results. This figure demonstrates how our modified equation Eq. (8) can minimize the difference of the curl and the magnitude of the divergence compared with Eq. (7). The vertical axes indicate the squared L2 norm of the difference of the curl (upper image) and the squared value of the divergence

Figure	grid size		size of interpolation region		T_{match}	T_{interp}	T_{sim}
	input	final	width	total			
Fig. 6	$192 \times 192 \times 256$	$192 \times 192 \times 256$	12	128×10^4	968	11	210
Fig. 7 (d)	$192 \times 192 \times 256$	$192 \times 192 \times 320$	4	$(192 \times 192 \times 4) \times 16$	–	7.5	280
Fig. 7 (g)	$192 \times 192 \times 256$	$240 \times 192 \times 256$	4	$(4 \times 192 \times 256) \times 12$	–	8.5	310
Fig. 8 (b)	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	21×10^4	90	1.4	65
Fig. 8 (d)	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	38×10^4	221	2.5	65
Fig. 8 (f)	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	41×10^4	257	3.0	65
Fig. 9	$256 \times 256 \times 64$	$256 \times 256 \times 64$	12	44×10^4	–	3.1	65
Fig. 10	$384 \times 384 \times 512$	$384 \times 384 \times 640$	8	$(384 \times 384 \times 8) \times 16$	–	100	3500
Fig. 1	$768 \times 256 \times 192$	$768 \times 764 \times 192$	8	$(768 \times 8 \times 192) \times 2$	–	15	6300

Table 1. Simulation statistics. T_{match} and T_{interp} are computation times corresponding to the frame matching and flow interpolation, respectively. T_{sim} is measured by running a fluid simulation with the grid size same as the final results. T_{match} is measured in seconds, and T_{interp} and T_{sim} are measured in seconds per frame.

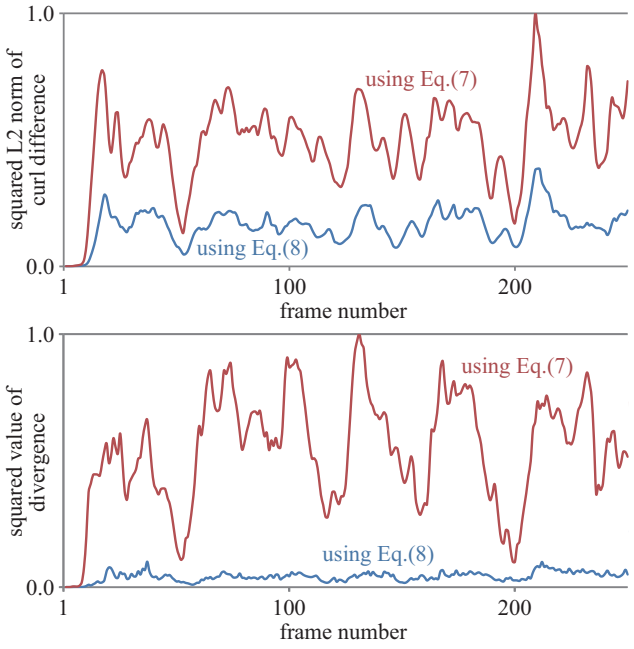

Fig. 5. A temporal change in the curl and divergence term for synthesized results. In this experiment, we use a single velocity field $\mathbf{u}_e(i)$ as an input and delete a part of it to use as an interpolation region. The squared L2 norm of curl difference and squared value of divergence are computed as $\sum_\Omega ||\nabla \times \mathbf{u}_\Omega(i) - \nabla \times \mathbf{u}_e(i)||^2$ and $\sum_\Omega (\nabla \cdot \mathbf{u}_\Omega(i))^2$, respectively.

(lower one) for the combined velocity fields, and the horizontal axis is the frame number. The vertical axes are normalized so that the maximum value of the red curve becomes one in each graph. In this experiment, we use a single velocity field $\mathbf{u}_e$ as an input and delete a part of it. Then the velocities in the deleted region are reconstructed by interpolation using either Eq. (8) or Eq. (7). The blue and red curves correspond to Eq. (8) and Eq. (7), respectively. Each value of the upper and lower figure is computed as $\sum_\Omega ||\nabla \times \mathbf{u}_\Omega - \nabla \times \mathbf{u}_e||^2$ and $\sum_\Omega (\nabla \cdot \mathbf{u}_\Omega)^2$, respectively. When using Eq. (7), the velocities are interpolated so that the magnitudes of their curls are minimized. Thus, the curl of the resulting velocity field is different from the curl of the original velocity field, $\mathbf{u}_e$. By contrast, our modified equation Eq. (8) takes into account the momentum equation by trying to preserve the curl computed from the velocity field at the previous time step. Consequently, the difference between the curls of the reconstructed field and the original field becomes much smaller than that obtained using Eq. (7) (see the upper figure). Furthermore, the divergence also becomes smaller when using our modified energy function (see the lower figure). This experiment implies that the energy function is better minimized by using the difference from the velocity field obtained by advecting the incompressible velocity fields (inputs) than by using the magnitude of the curl.

By introducing a relative weighting coefficient to either term in Eq. (8), the influence of the curl or divergence terms can be controlled. That is, by using a large coefficient for the divergence term, our method would minimize the divergence term more preferentially than the curl term. However, using too big/small coefficient could lead to unnatural and implausible velocities in the results. Furthermore, we cannot set the coefficient to zero or ∞. The curl and divergence components of a velocity field are orthogonal, and one component lies in a non-trivial null space of the other component [Ando et al. 2015]. Therefore, an infinite number of vector fields could have the same curl or divergence, thus Eq. (9) cannot be solved uniquely with only one term.

4 RESULTS

Figs. 6 - 10 and 1 show the results created using our method. We used a desktop PC with an Intel Core i7-6700K CPU and NVIDIA GeForce GTX 1080 and 32GB of memory to compute all the examples. The grid sizes, the parameters, and the computation times are summarized in Table 1. Our method is much faster than the full simulation with a grid size same as our final result (see T_{interp} and T_{sim}). The orange spheres indicate obstacles in Figs. 6 and 8. To render the 3D results shown in Figs. 6 - 9, we used the physically-based renderer "Mitsuba" [Jakob 2010]. The videos corresponding to these examples can be found in the supplementary material.

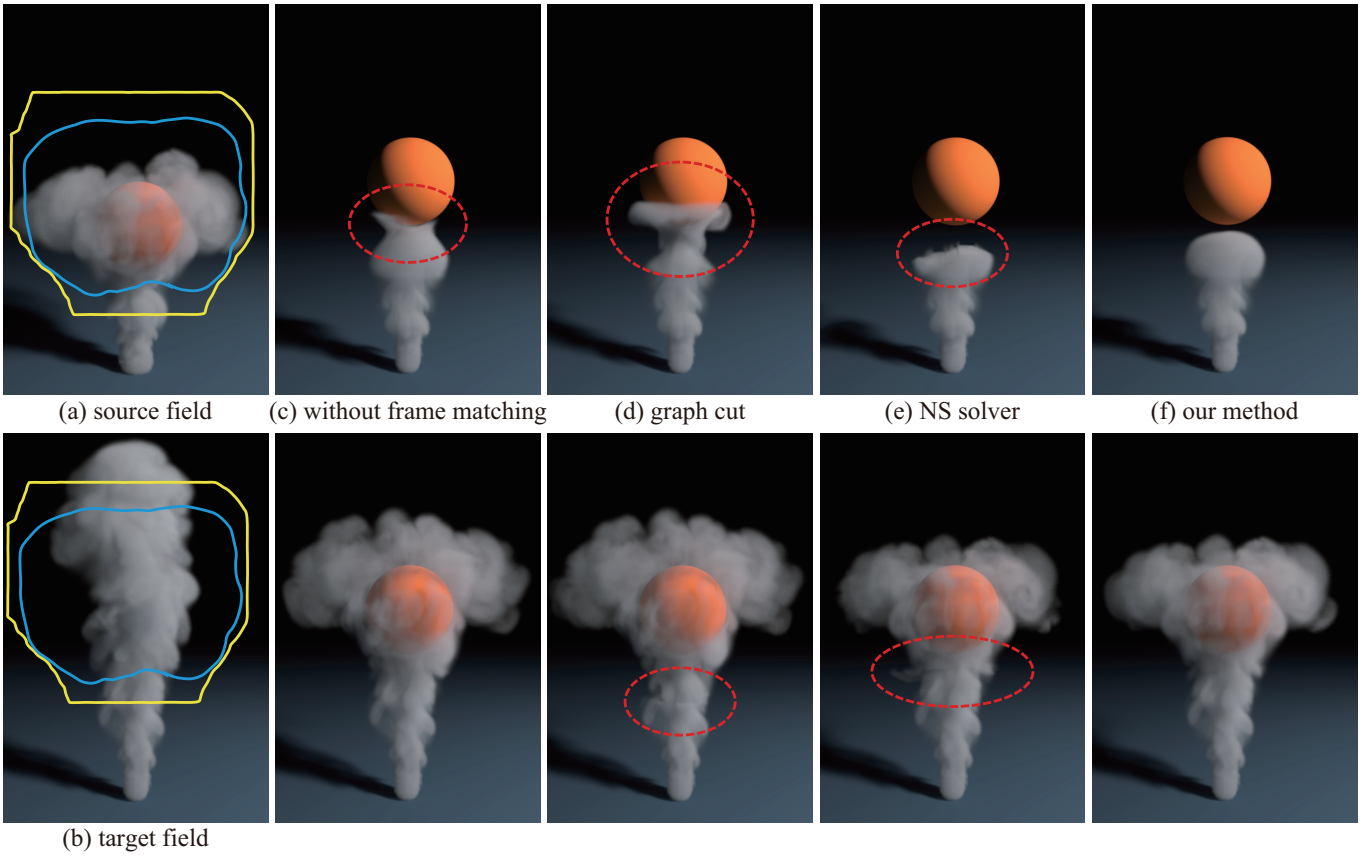

Fig. 6. A comparison of results with our method. In (c), distributions of input flows are not matched, so visual artifacts are visible. In (d), we apply graph cut for obtaining the interpolation region in order to realize smoother synthesis. However, velocity fields are only copied in graph cut, so discontinuities occur. (e) is created by solving the Navier-Stokes equations in the interpolation region with the boundary velocities pinned to the input velocity fields, but the smoke stays around there. By contrast, our results (f) can be smoothly combined and such artifacts are not visible.

Fig. 6 shows a comparison of the results obtained using different interpolation schemes. Figs. 6 (a) and (b) are the input flows, (a) is rising smoke with a spherically shaped obstacle, and (b) is rising smoke only. The grid size of these two inputs is the same. The blue curves indicate the user specified region and the copied region, the same as in Fig. 2. The synthesis region is specified by a 2D curve, and the region in the depth direction is the same as the grid size for the input. For comparison, the yellow curves are calculated by applying a graph cut [Kwatra et al. 2003] to the difference between the two input velocity fields, used for the result (d). To apply the graph cut, grid points on the boundary of the user specified region (on the blue curve) are connected to an edge with infinite value. Figs. 6 (c) - (e) are created using different schemes, and Fig. 6 (f) is our result.

Fig. 6 (c) is created using our method, but the frame matching process is not applied. Therefore, there are velocities from the source field in the user specified region before the smoke reaches there, and the smoke is stretched inside the red dotted curve. By contrast, in Fig. 6 (f), because the distributions of the velocities in the user specified region become similar by using the frame matching, such problems do not occur. The temporal offset in our result is $i_{ofs} = 36$.

Fig. 6 (d) is generated by copying the velocity fields from (a) to (b) inside the yellow curve calculated using a graph cut. Although the graph cut algorithm finds the interpolation region by minimizing the difference in flow at the boundaries, artifacts clearly appear. That is, since the physical properties are not guaranteed near the boundary of the synthesis region, the smoke density disappears in the region within the red dotted curve. Furthermore, since the temporal similarity between the input flows is not taken into account, the smoke is stretched inside red dotted curve in the upper image of (d). However, such problems do not occur with our method; this demonstrates that taking the physical properties into account is important.

Fig. 6 (e) is created by solving the Navier-Stokes equations [Stam 1999] in the interpolation region, with the boundary velocities pinned to the input velocity fields, and frame matching is applied. In this example, visual artifacts arise in the region within the red dotted curve, because the velocities around the boundary of the interpolation region are not smooth. Since the projection term in

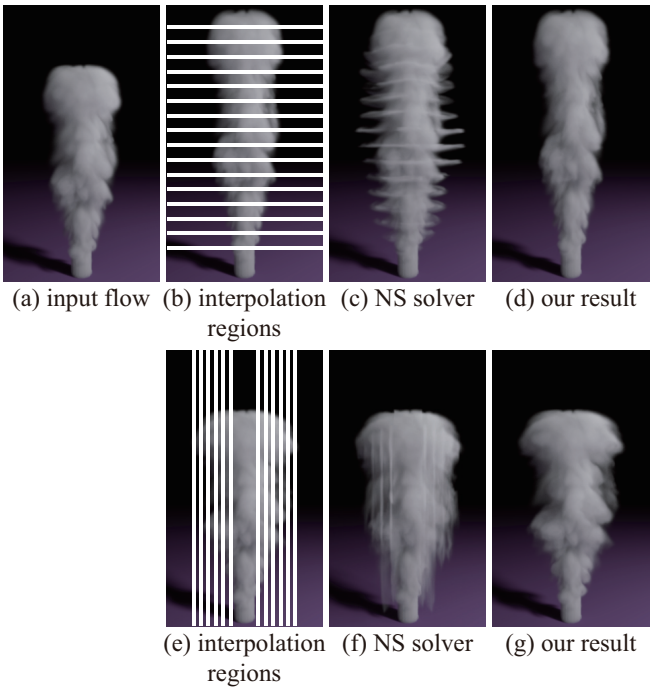

Fig. 7. Partially extending smoke animations using our method. We can realize the velocity field retargeting.

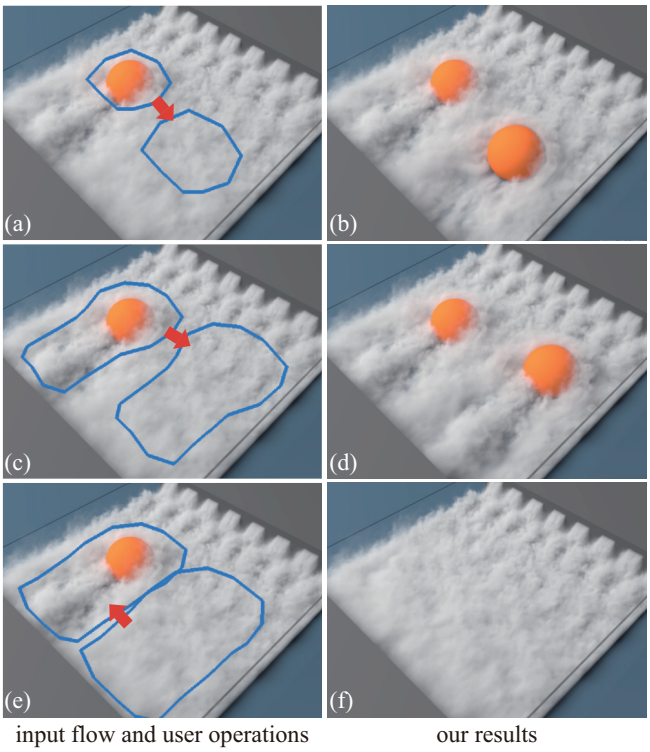

Fig. 8. Synthesizing various flows from a single input flow using a part of itself.

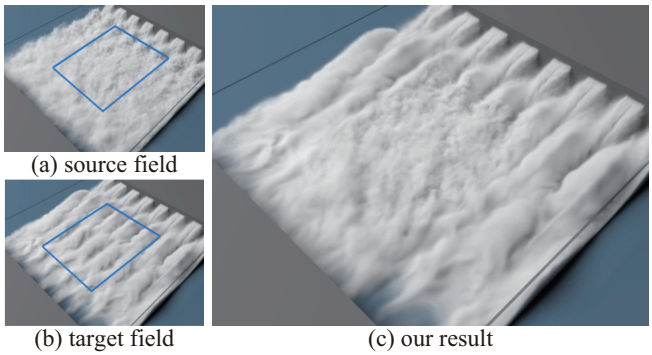

Fig. 9. Combining two input flows with different turbulent motions.

the Navier-Stokes equations is solved for a velocity field with such discontinuities, implausible velocities are generated around the interpolation region. As a result, the smoke remains around there. This result indicates that solving the Navier-Stokes equations is not suitable for interpolating velocity fields. Our method can successfully combine the input flows smoothly and naturally (Figs. 6 (f)). Our method is much more effective for interpolating velocity fields.

In Fig. 7, we create a flow extended from a single input flow. Fig. 7 (a) is the input flow. Figs. 7 (b) and (e) are the input flow and interpolation regions, shown by white rectangles. Figs. 7 (c) and (f) are results created by applying a Navier-Stokes solver [Stam 1999] to compute the flow in the white regions. Figs. 7 (d) and (g) are our results. We create a taller smoke plume in (b), (c) and (d), and wider smoke in (e), (f) and (g). In Figs. 7 (c) and (f), visual artifacts are observed around the interpolation regions. Although the interpolation regions are quite small, the advected velocities and velocities at the boundary of the interpolation region are certainly different. The NS solver needs several time steps to smoothly connect these different velocities. Until that time step, the velocities are discontinuous at the boundary, resulting in the artifacts. As shown in Figs. 7 (d) and (g), we can achieve velocity field retargeting [Rubinstein et al. 2008]. Especially, since the flow is extended without the center of the simulation space in (g), we can preserve the shape of the smoke around the source of the input flow. However, the smoke plume in Fig. 7 (d) slightly speeds up in the interpolation regions (see the accompanying video) because the frame matching process is not applied in this example. That is, the same problem as in Fig. 6 (c) actually occurs in this case.

Fig. 8 shows the various results edited from a single flow by using a part of itself. Left images in Fig. 8 show the user's operations and right ones show the corresponding results. The temporal offset in these results is i_{ofs} = (b) 4, (d) −6, (f) 0. In the top images (a) and (b), the user copies the region around the obstacle to the region indicated by the red arrow. In this case, only the flow around the obstacle is smoothly combined but the obstacle removes the density at the copied region, resulting in a natural density distribution even

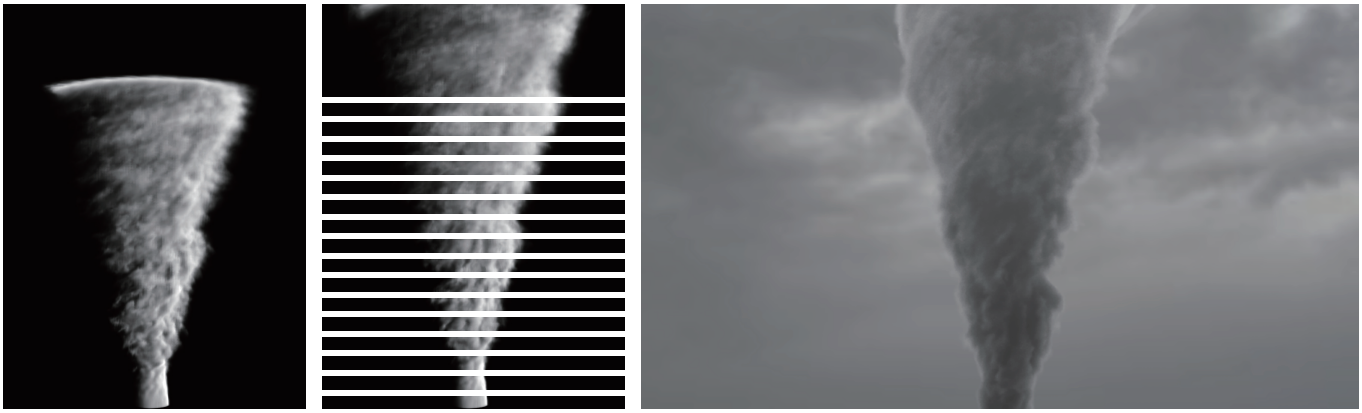

Fig. 10. A partially extended tornado animation (right) created by interpolating white regions (center) from a single input animation (left).

behind the obstacle. In the center images (c) and (d), the user copies the obstacle together with the flow behind it. Then, the flow behind the obstacle is smoothly combined too, resulting in a more natural flow. Next, in the bottom images (e) and (f), the user uses our method to remove the obstacle. The obstacle is successfully removed and a plausible flow is created.

In Fig. 9, we combine two flows with different turbulent motions. Figs. 9 (a) and (b) show the input flows: the flow in (a) is turbulent but the flow in (b) does not have small-scale turbulent motion. (c) is the combined result created using these two inputs. This example demonstrates that our method has the ability to combine flows even when input flows have largely different turbulent motions.

In Fig. 10, we create a tornado animation extended from a single input animation. Fig. 10 (a) is the input flow. Fig. 10 (b) is the extended flow and interpolation regions are shown by white rectangles. Fig. 10 (c) is our result; we create taller tornado. As shown in this result, we can also achieve the retargeting for such large-scale flow.

In Fig. 1, we create a large-scale sandstorm animation from two input flows simulated in a small-scale area. Fig. 1 (a) shows the two input flows. Fig. 1 (c) shows the result created by placing the input flows at multiple positions and interpolating the white regions as shown in Fig. 1 (b). This example demonstrates that our method can be used to create a large-scale flow by combining multiple small-scale flows. Since Poisson's equation is solved within only a small fraction of the total area, the memory consumption is much smaller than a full simulation.

5 APPLICABILITY AND LIMITATIONS

Currently, our method has some limitations. We discuss these in this section.

Our method is the most effective for combining velocity fields in which the velocities have similar directions and magnitudes. Even if largely different velocity fields are to be combined, our method can compute a smooth flow, but the divergence may not be small enough, and visual artifacts might occur. As it is difficult to explicitly define the region of applicability, we present some tests shown in Figs. 11 - 13 to illustrate cases in which our method fails to yield good results. The animations corresponding to these experimental examples can be found in the supplemental video.

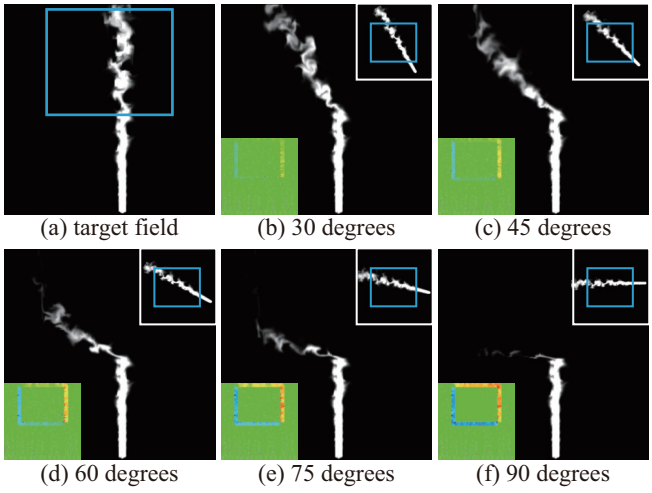

Fig. 11. Experimental results combined two velocity fields which have the different direction of velocities. (a) shows the target field. (b) through (f) show the results of the combined flows; the inset images in each result are the source field (the top right corner) and the divergence of each combined velocity field (the bottom left corner). The divergence is visualized using a pseudo color. The difference of directions between target and source flow is indicated by the caption of each figure.

Fig. 11 shows the experimental results where the velocities in two combined velocity fields are in different directions. Fig. 11 (a) is the target field: a fixed boundary condition with an upward vertical velocity is applied to the velocity field at the bottom of the simulation space. A certain amount of smoke density is added at the bottom center of each frame for visualizing the flow. The blue rectangle indicates the user specified region. Figs. 11 (b) - (f) are the results of the combined flows; the inset images in each result are the source

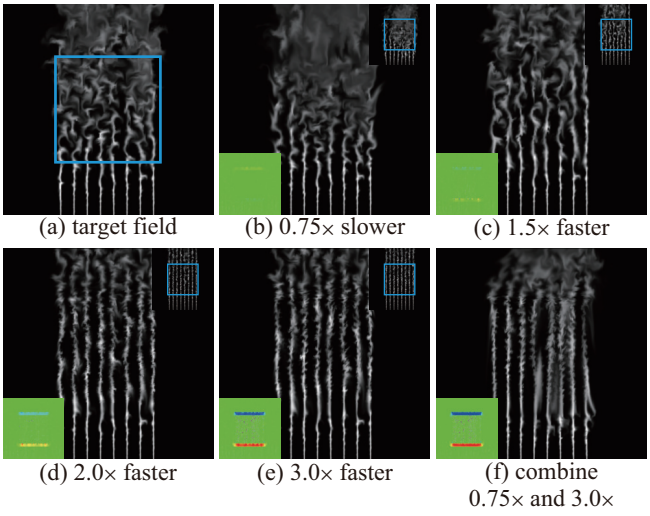

Fig. 12. Experimental results combined two velocity fields which have the different magnitude of velocities. (a) shows the target field. (b) through (e) are the combined results; the inset images in each result are the source field (the top right corner) and the divergence of each combined velocity field (the bottom left corner). The caption of each result indicates a difference of the speed of the source field from the target field. (f) is the result created by combining the very slow flow (source for (b)) and the very fast flow (source for (e)).

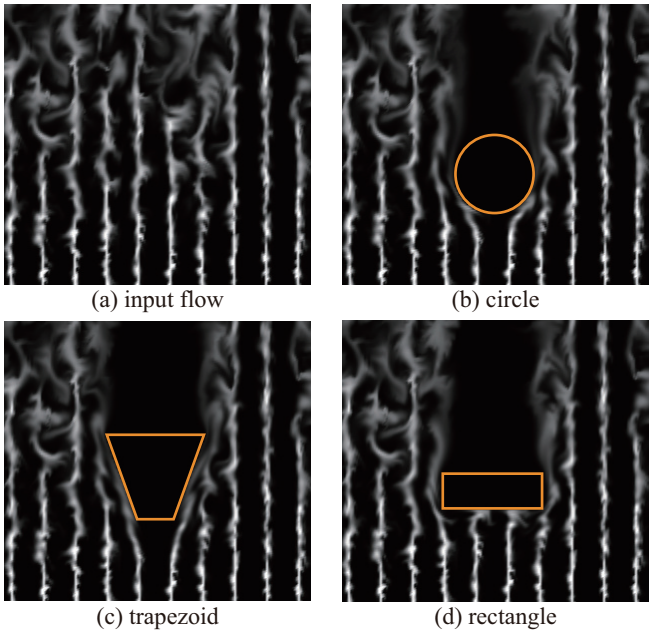

Fig. 13. Experimental results combined the velocity field and the stationary field. The orange curves indicate regions for the stationary fields.

field (the top right corner) and the divergence of each combined velocity field (the bottom left corner) visualized using a pseudo color (blue, green, and red indicate the minimum, zero, and the maximum values of the divergence). Each source field is obtained by rotating the target field (a) around its center, and the angle of rotation is shown in the caption of each figure. As shown in these results, we found that the divergence of the interpolated flow is sufficiently small as long as the difference in the direction of the overall flow is smaller than about 45 degrees. Otherwise, the divergence is not small, so the smoke disappears around the interpolation region, even though the resulting velocity field is continuous.

Fig. 12 shows the experimental results where two velocity fields with two different speeds are combined. Fig. 12 (a) is the target field: an upward vertical velocity is added at the bottom of the simulation space. The blue rectangle indicates the user specified region. Figs. 12 (b) - (e) are the combined results; the inset images in each result are the source field (the top right corner) and the divergence of each combined velocity field (the bottom left corner). The caption of each result indicates a difference between the magnitudes of the target (a) and source (b) - (e) fields. Fig. 12 (f) is created by combining the source field for (b) (very slow flow) and (e) (very fast flow). When the difference in speed is relatively small as in Figs. 12 (b) and (c), the divergence is small enough to produce natural and plausible flow. In Figs. 12 (d) and (e), unnatural flow is observed around the interpolation regions due to the large difference in the flow speeds. Serious artifacts appear when we try to combine velocity fields with significant speed differences (see Fig. 12 (f)).

Next, Fig. 13 shows the experimental results obtained by combining a velocity field and a stationary field. Fig. 13 (a) is an input flow where the smoke flows vertically. Figs. 13 (b) - (d) are our results combined the input flow and the stationary fields indicated by the orange curves. The interpolation regions are around the outside of the orange curve, and their widths are 20, 16, and 16 for (b), (c), and (d), respectively. Our method can generate nice-looking velocity fields flowing smoothly along the boundaries of the shapes of the stationary field when the direction of the input flow is not perpendicular to the stationary field, as shown in Figs. 13 (b) and (c). However, since the velocity field is not perfectly divergence-free, the smoke density might decrease. The situation is worse when the input flow is perpendicular to the stationary field as shown in Fig. 13 (d); the interpolated velocity field does not flow along the boundary of the stationary field and, as a result, the smoke density disappears at the boundary. From this experiment, we found that plausible flow cannot be interpolated if there are obstacles with many surfaces perpendicular to the flow. Otherwise, plausible flow seems to be generated by our interpolation method. One solution to this problem is to specify some additional constraints, such as the direction and magnitude, for interpolating the flow. This is a challenging but interesting problem; we leave this task for our future work.

The user specified width of the interpolation region has a big effect in our method. If the width is very small, the divergence becomes larger and visual artifacts could appear. By contrast, if the width is very large, the divergence is close to zero but the difference of the curl becomes larger. The width of the interpolation region needs to be chosen appropriately; we currently determine it by a trial-and-error process. However, in our experiments, we found that

such problems do not occur if the width is set to about 2 – 7% of the entire simulation space.

Our combined velocity fields might have discontinuities at the boundaries of the interpolation regions if the input velocity fields have large differences, e.g., combining wide smoke with thin smoke. In this case, visual artifacts are found because unexpected velocities are generated in the interpolated regions in order to minimize the magnitude of the divergence. This problem could be addressed by applying a deformation method for fluid animations (e.g., [Sato et al. 2015; Thuerey 2016]) in order to stitch the gap between the distributions around the interpolation region.

Our method focuses on smoke simulations calculated on a uniform grid. In our results (Sec.4), we combined velocity fields which have the same grid interval. Our method can be applied to velocity fields with different grid intervals, or even on a non-uniform grid, by resampling the flow on a uniform grid with the same grid interval. However, we might have to carefully choose the resampling scheme in order to avoid over-smoothing artifacts.

In Sec. 4, we showed that our method can combine multiple flows and our method is faster than full simulations. Furthermore, since multiple synthesis regions can be computed independently, the method is suitable for parallel computing. However, the time for loading input velocity fields is taken at every frame. For our results (Sec. 4), about 0.35 – 1.0 seconds per frame were needed to load the single 3D velocity field. If we can pre-load all the input data, this loading time can be shortened but a huge memory capacity is required for storage.

Our approach is quite efficient for synthesizing a large-scale flow as shown in Fig. 10 and 1. As described in Sec. 4, even if we synthesize such a large-scale flow, the memory consumption is low for our interpolation process because we need only the velocities around the interpolation region. However, unfortunately, for the advection of the density, we need two huge volumes (velocity and density) to visualize the flow. To reduce such memory cost, we need to break up the domain, or perform it out-of-core in stages.

We tried different ways of measuring the similarity in the frame matching process, such as the simple density difference or the one used for the alpha mask in the LazyFluids method [Jamriska et al. 2015]. However, failures are sometimes observed: minimizing the density differences might reduce the number of overlapping frames. Therefore, we count similar grids and maximize the sum of the total number of such grids so that the number of overlapping frames is increased. However, our frame matching method sometimes fails to find good matching frames, for example, when the smoke regions in Ω_{usr} are quite small. We believe that our current approach works better than using simple density differences but it is not perfect; the method is heuristically derived and there might be better solutions. We need further investigation to find the best solution to this problem.

6 CONCLUSIONS

We have proposed a method for combining existing fluid animations while considering the physical laws of the fluid. Our method smoothly combines multiple flows by solving a minimization problem. For the interpolated flow, our framework can take into account the inviscid, incompressible Navier-Stokes equations. Using our method, natural and plausible flows can be efficiently and easily created by combining existing flow data.

For future work, we are planning to extend our method to combine velocity fields with large differences, as described in Sec. 5. This might be achieved by partially deforming the velocity fields. We are also planning to extend our method to other types of fluid simulations such as liquid and fire simulations. In addition, we will reduce the computation and storage costs by extending our method to adaptive grids.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Hiroyuki Ochiai at Kyushu University and Prof. Chris Wojtan at IST Austria for their valuable comments. The authors would also like to thank the reviewers for their constructive comments to improve our paper. This work was supported by JSPS KAKENHI Grant Number JP15H05924.

REFERENCES

Ryoichi Ando, Nils Thuerey, and Chris Wojtan. 2015. A stream function solver for liquid simulations. *ACM Transactions on Graphics* 34, 4 (2015), Article 53.

H. Bhattacharya, M. B. Nielsen, and R. Bridson. 2012. Steady state Stokes flow interpolation for fluid control. In *Short Paper Proceedings of Eurographics 2012*. 57–60.

Morten Bojsen-Hansen and Chris Wojtan. 2016. Generalized non-reflecting boundaries for fluid re-simulation. *ACM Transactions on Graphics* 35, 4 (2016), Article 96.

Robert Bridson. 2015. *Fluid simulation for computer graphics*. CRC Press.

R. Fattal and D. Lischinski. 2004. Target-driven smoke animation. *ACM Transactions on Graphics* 23, 3 (2004), 439–446.

R. Fedkiw, J. Stam, and H. W. Jansen. 2001. Visual Simulation of Smoke. In *Proceedings of ACM SIGGRAPH 2001*. 15–22.

B. E. Feldman, J. F. O'Brien, and O. Arikan. 2003. Animating Suspended Particle Explosions. In *Proceedings of ACM SIGGRAPH 2003*. 708–715.

Matthew Fisher, Peter Schroder, Mathieu Desbrun, and Hugues Hoppe. 2007. Design of tangent vector fields. *ACM Transactions on Graphics* 26, 3 (2007), Article 56.

N. Foster and R. Fedkiw. 2001. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*. 23–30.

N. Foster and D. Metaxas. 1996. Realistic animation of liquids. *Graphical Models and Image Processing* 58, 5 (1996), 471–483.

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.

Wenzel Jakob. 2010. Mitsuba renderer. (2010). http://www.mitsuba-renderer.org.

O. Jamriska, J. Fiser, P. Asente, J. Lu, E. Shechtman, and D. Sykora. 2015. LazyFluids: appearance transfer for fluid animations. *ACM Transactions on Graphics* 34, 4 (2015), Article 92.

Theodore Kim and John Delaney. 2013. Subspace fluid re-simulation. *ACM Transactions on Graphics* 32, 4 (2013), Article 62.

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics* 22, 3 (2003), 277–286.

A. McNamara, A. Treuille, Z. Popovic, and J. Stam. 2004. Fluid control using the adjoint method. *ACM Transactions on Graphics* 23, 3 (2004), 449–456.

R. Miyazaki, Y. Dobashi, and T. Nishita. 2002. Simulation of cumuliform clouds based on computational fluid dynamics. In *Proceedings of EUROGRAPHICS 2002 Short Presentations*. 405–410.

D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. 2002. Physically based modeling and animation of fire. *ACM Transactions on Graphics* 21, 3 (2002), 721–728.

Michael B. Nielsen and Brian B. Christensen. 2010. Improved Variational Guiding of Smoke Animations. *Computer Graphics Forum* 29, 2 (2010), 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* 32, 6 (2013), Article 184.

P. Perez, M. Gangnet, and A. Blake. 2003. Poisson image editing. *ACM Transactions on Graphics* 22, 3 (2003), 313–318.

Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. 2014. Blending liquids. *ACM Transactions on Graphics* 33, 4 (2014), Article 137.
Michael Rubinstein, Ariel Shamir, and Shai Avidan. 2008. Improved seam carving for video retargeting. *ACM Transactions on Graphics* 27, 3 (2008), Article 16.
S. Sato, Y. Dobashi, and T. Nishita. 2016. A combining method of fluid animations by interpolating flow fields. In *Proceedings of SIGGRAPH Asia 2016 Technical Briefs*. Article 4.
S. Sato, Y. Dobashi, Y. Yue, K. Iwasaki, and T. Nishita. 2015. Incompressibility-preserving deformation for fluid flows using vector potentials. *The Visual Computer* 31, 6 (2015), 959–965.
Jos Stam. 1999. Stable Fluids. In *Proceedings of ACM SIGGRAPH 1999, Annual Conference Series*. 121–128.
Nils Thuerey. 2016. Interpolations of Smoke and Liquid Simulations. *ACM Transactions on Graphics* 36, 1 (2016), Article 3.
N. Thürey, R. Keiser, M. Pauly, and U. Rüde. 2006. Detail-preserving fluid control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 7–12.
A. Treuille, A. McNamara, Z. Popovic, and J. Stam. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics* 22, 3 (2003), 716–723.
Ke Wang, Weiwei, Yiying Tong, Mathieu Desbrun, and Peter Schröder. 2006. Edge subdivision schemes and the construction of smooth vector fields. *ACM Transactions on Graphics* 25, 3 (2006), 1041–1048.
G. D. Yngve, J. F. O'Brien, and J. K. Hodgins. 2000. Animating explosions. In *Proceedings of ACM SIGGRAPH 2000*. 29–36.

APPENDIX

Derivation of Eq. (6), (9) from Eq. (7), (8).

Let $E = ||\nabla \times \mathbf{u}||^2 + (\nabla \cdot \mathbf{u})^2$. Substituting $\mathbf{u} + \delta\mathbf{u}$ into E instead of $\mathbf{u}$, we put E to be $E + \delta E$;

$$\begin{aligned}\delta E &= ||\nabla \times (\mathbf{u} + \delta\mathbf{u})||^2 + (\nabla \cdot (\mathbf{u} + \delta\mathbf{u}))^2 - E \\ &= ||\nabla \times \mathbf{u} + \nabla \times \delta\mathbf{u}||^2 + (\nabla \cdot \mathbf{u} + \nabla \cdot \delta\mathbf{u})^2 - E \\ &= 2\langle \nabla \times \delta\mathbf{u}, \nabla \times \mathbf{u}\rangle + 2(\nabla \cdot \delta\mathbf{u})(\nabla \cdot \mathbf{u}) \\ &\quad + ||\nabla \times \delta\mathbf{u}||^2 + (\nabla \cdot \delta\mathbf{u})^2.\end{aligned}$$

Then, we use the following formulas of vector calculus;

$$\langle \nabla \times \delta\mathbf{u}, \nabla \times \mathbf{u}\rangle - \langle \delta\mathbf{u}, \nabla \times (\nabla \times \mathbf{u})\rangle = \nabla \cdot (\delta\mathbf{u} \times (\nabla \times \mathbf{u})),$$
$$(\nabla \cdot \delta\mathbf{u})(\nabla \cdot \mathbf{u}) + \langle \delta\mathbf{u}, \nabla(\nabla \cdot \mathbf{u})\rangle = \nabla \cdot (\delta\mathbf{u}(\nabla \cdot \mathbf{u})).$$

Since $\delta\mathbf{u}$ is sufficiently small, we interpret divergent terms (right hand side) in both above equations equal 0 from the Gauss divergence theorem. Then, δE can be rewritten as

$$\begin{aligned}\delta E &= 2\langle \delta\mathbf{u}, \nabla \times (\nabla \times \mathbf{u})\rangle - 2\langle \delta\mathbf{u}, \nabla(\nabla \cdot \mathbf{u})\rangle \\ &\quad + ||\nabla \times \delta\mathbf{u}||^2 + (\nabla \cdot \delta\mathbf{u})^2 \\ &= -2\langle \delta\mathbf{u}, -\nabla \times (\nabla \times \mathbf{u}) + \nabla(\nabla \cdot \mathbf{u})\rangle \\ &\quad + ||\nabla \times \delta\mathbf{u}||^2 + (\nabla \cdot \delta\mathbf{u})^2.\end{aligned}$$

Hence, the first variation of E with respect to $\delta\mathbf{u}$ is

$$-\nabla \times (\nabla \times \mathbf{u}) + \nabla(\nabla \cdot \mathbf{u}).$$

From the above proof, the derivative of Eq. (7) with respect to $\mathbf{u}_\Omega(i)$ so that Eq. (7) is minimized is expressed by Eq. (6). Similarly, when we put $\nabla \times \mathbf{u}$ to be $\nabla \times \mathbf{u} - \nabla \times \hat{\mathbf{u}}$, we have the derivative of Eq. (8) equals Eq. (9).