A Combining Method of Fluid Animations by Interpolating Flow Fields

Syuhei Sato*
UEI Research

Yoshinori Dobashi
Hokkaido University / UEI Research

Tomoyuki Nishita
UEI Research / Hiroshima Shudo University

Abstract

The computational cost for creating realistic fluid animations by simulation is generally very expensive. In digital production environment, existing precomputed fluid animations are often reused for different scenes in order to reduce the cost for creating scenes containing fluids. However, applying same animations to different scenes produces unacceptable results, so the animation needs to be edited. In order to do this, we develop a method for synthesizing desired flow fields by combining existing flow fields. Our system allows the user to place existing flow fields at arbitrary positions, and combine them by interpolating the regions between these flow fields, to synthesize a new flow field. The interpolation of the flow fields is realized by solving a minimization problem. Our minimization problem consists of two energy functions for smoothly interpolating the velocities and satisfying the incompressibility. Our method can create the desired incompressible flow fields by reusing existing flow fields.

Keywords: fluid simulation, reusing existing fluid animations, incompressibility, interpolating velocity fields

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

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 its 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. We can avoid the costly simulation by reusing precomputed, existing fluid animations. However, the same simulation results cannot be used for different scenarios; we need to modify/edit the existing fluid animations so that they fit the different scenes, such as when adding obstacles and changing the flow directions. However, such drastic modifications require re-simulation of the fluids, which is time-consuming.

In this paper, we develop a method for synthesizing new flow fields by combining existing flow fields. Our method provides a tool for seamless editing and cloning of selected regions of existing flows. The user can crop arbitrary regions from the existing flow fields and paste them at arbitrary positions in the synthesis region where the new flow field is to be created. The multiple flow fields cropped from the existing flow fields are then seamlessly combined by interpolating the flows at the boundaries of the cropped regions. For this interpolation, we solve a minimization problem at the boundary regions. We minimize an energy function consisting of two terms; one is a sum of gradients of velocities across the boundary regions and the other is defined as L2 norm of a divergence of velocities. The second term is used for satisfying the incompressibility of the flow, which is an important physical property of fluids.

We focus on grid-based fluid simulations. Our method can combine existing flow fields, so new desired flow fields can be created efficiently and easily.

*e-mail:syuhei_sato@dwango.co.jp

2 Related Work

Stam developed an unconditionally stable solver [Stam 1999] for the Navier-Stokes equations. Since this method was introduced, many methods have been proposed for simulating various fluid phenomena [Bridson 2008; Fedkiw et al. 2001; Feldman et al. 2003; Foster and Fedkiw 2001; Miyazaki et al. 2002; Nguyen et al. 2002; Yngve et al. 2000]. Although these methods can produce realistic fluid animation, the user must repeatedly execute fluid simulations with different parameter settings until the desired fluid animation is created.

Several procedural methods have been proposed for efficiently creating fluid-like animations. For example, some methods for editing fire animations were presented that create the desired fire animations at low cost [Fuller et al. 2007; Lamorlette and Foster 2002]. Pighin et al. [Pighin et al. 2004] proposed a method for editing simulated flow fields by introducing advected radial basis functions. Since these methods do not take into account physical laws such as the incompressibility of the fluid, unrealistic results can be produced.

To efficiently simulate fluids, model reduction methods [Treuille et al. 2006; Wicke et al. 2009] were developed. These methods prepare many sets of velocity fields obtained by simulating fluids with various parameter settings and initial conditions. Then basis functions are generated by applying principal component analysis (PCA) to the precomputed set of velocity fields. At the cost of expensive precomputation, the flow field can be updated efficiently by computing the Navier-Stokes equations in the basis space. Kim and Delaney [Kim and Delaney 2013] presented a method for efficient re-simulation with different parameter settings. This method generates basis functions by applying 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 these methods, the flow fields that can be created are limited to those represented by a linear combination of the basis functions. Furthermore, the memory consumption in applying PCA can be a bottleneck for a large database. Bojsen-Hansen and Wojtan [Bojsen-Hansen and Wojtan 2016] introduced non-reflecting boundary conditions with inflow/outflow constraints for fluid re-simulation. By using this method, the user can locally re-simulate a fluid animation as a post-process. However, as shown in the results of [Bojsen-Hansen and Wojtan 2016], only rectangular domains are used for the re-simulation. In contrast, since the user can specify an arbitrary shape for synthesis regions, our method can edit fluid animations with a higher degree of freedom. In addition, fluid simulations do not have to be executed to create new flow fields, because we use existing precomputed fluid animations.

Several methods have been proposed for efficiently creating new flow fields by reusing existing fluid animations. Raveendran et al. developed a method for smoothly blending existing fluid animations [Raveendran et al. 2014]. This method semi-automatically matches two existing liquid animations to plausibly interpolate the input animations. By using stream functions and vector potentials, Sato et al. showed that 2D and 3D flow fields could be deformed while guaranteeing the incompressibility of the flow [Sato et al. 2014; Sato et al. 2015]. Using these methods, the desired flow fields can be created efficiently by reusing existing flow fields. However, drastic changes in flow fields are difficult to achieve using these

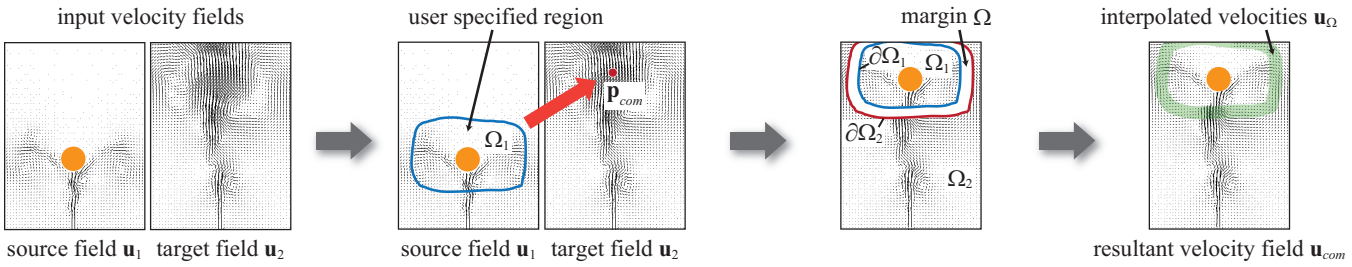


Figure 1: *Overview of our method. For the sake of visual clarity, each 3D velocity field is illustrated as a 2D velocity field.* $\mathbf{p}_{com}$ *is a user specified position where* Ω_1 *is placed in target field,* $\partial\Omega_1$ *and* $\partial\Omega_2$ *are boundaries of* Ω.

deformation methods. In contrast, our method achieves relatively greater changes in flow by partially combining multiple flow fields.

3 Our Method

Fig. 1 shows an overview of our method. First, input velocity fields $\mathbf{u}(t)$ are prepared by solving incompressible Navier-Stokes equations, where $t = 0, 1, \cdots, T-1$ represents the frame count, and T is the number of frames of the input velocity field. For the sake of clarity, we describe our method by using two input velocity fields $\mathbf{u}_1(t)$ and $\mathbf{u}_2(t)$ (see Fig. 1). These two flow fields are combined in the following way. Let us call the two fields, $\mathbf{u}_1(t)$ and $\mathbf{u}_2(t)$, the source and target fields, respectively. In the source field, the user specifies an arbitrary region Ω_1 (the region surrounded by the blue curve in Fig. 1). Then, the user specifies a position $\mathbf{p}_{com}$ in the target field (the red point in Fig. 1), and the flow in Ω_1 in the source field is copied into the target field so that the center of Ω_1 coincides with $\mathbf{p}_{com}$. A margin, Ω, with a predefined width is automatically created around the boundary of Ω_1 as shown in Fig. 1. The region in the target-field without Ω_1 and Ω is denoted by Ω_2 (Fig. 1), and the boundaries between Ω and Ω_1, Ω and Ω_2 are denoted by $\partial\Omega_1$, $\partial\Omega_2$ (blue and red curves Fig. 1), respectively. In order to generate a smooth and continuous flow field, the velocities $\mathbf{u}_\Omega(t)$ in the region Ω are calculated by minimizing an energy function $E_\Omega(t)$. Our energy function $E_\Omega(t)$ comprises two elements: one for smoothly interpolating the velocities and the other for satisfying the incompressibility. The definition of the energy function is given in Sec. 4. Finally, we obtain the resultant velocity field $\mathbf{u}_{com}(t)$: $\mathbf{u}_{com}(t) = \mathbf{u}_\Omega(t)$ at Ω, $\mathbf{u}_{com}(t) = \mathbf{u}_1(t)$ at Ω_1, and $\mathbf{u}_{com}(t) = \mathbf{u}_2(t)$ at Ω_2. To visualize the flow, smoke densities are advected by the resultant velocity field $\mathbf{u}_{com}(t)$.

4 Definition of Energy Functions

In this section, we define the energy function used in the minimization problem. Our energy function $E_\Omega(t)$ consists of two elements, $E_{grd}(t)$ and $E_{div}(t)$. $E_{grd}(t)$ is an energy function for smoothly interpolating the velocities in the margin Ω, and is given by the following equation.

$$E_{grd}(t) = \sum_\Omega ||\nabla \mathbf{u}_\Omega(t)||^2, \tag{1}$$

where ∇ is the gradient operator. We introduce Eq.(1), so that we can compute the velocities $\mathbf{u}_\Omega(t)$ such that the gradients of these are as small as possible. This term is used to remove discontinuities in the resulting flow. The incompressibility of the flow is taken into account by the second term, $E_{div}(t)$, which is defined as the following equation.

$$E_{div}(t) = \sum_\Omega ||\nabla \cdot \mathbf{u}_\Omega(t)||^2, \tag{2}$$

where $\nabla\cdot$ is the divergence operator. Then, our energy function $E_\Omega(t)$ is defined as the sum of the above two energy functions.

$$\begin{aligned} E_\Omega(t) &= E_{grd}(t) + \alpha E_{div}(t) \\ &= \sum_\Omega \{||\nabla \mathbf{u}_\Omega(t)||^2 + \alpha ||\nabla \cdot \mathbf{u}_\Omega(t)||^2\}, \end{aligned} \tag{3}$$

where α is a coefficient for adjusting the influence of $E_{div}(t)$. To obtain $\mathbf{u}_\Omega(t)$, we solve the following minimization problem

$$\arg\min_{\mathbf{u}_\Omega(t)} \sum_\Omega \{||\nabla \mathbf{u}_\Omega(t)||^2 + \alpha ||\nabla \cdot \mathbf{u}_\Omega(t)||^2\}, \tag{4}$$

and we set the boundary conditions $\mathbf{u}_\Omega(t) = \mathbf{u}_1(t)$ at $\partial\Omega_1$ and $\mathbf{u}_\Omega(t) = \mathbf{u}_2(t)$ at $\partial\Omega_2$. By taking the derivative of Eq.(4) with respect to $\mathbf{u}_\Omega$, we obtain

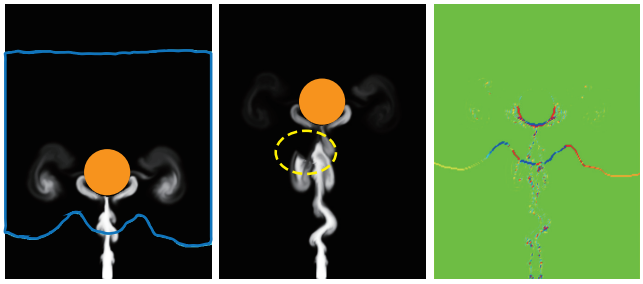
$$\nabla^2 \mathbf{u}_\Omega(t) + \alpha \nabla(\nabla \cdot \mathbf{u}_\Omega(t)) = \mathbf{0}, \tag{5}$$

where ∇^2 is the Laplace operator. We obtain a numerical solution to Eq.(5), we used the biconjugate gradient stabilized method (BiCGSTAB).

5 Results

Figs. 2 - 6 show results created using our method. We used a desktop PC with an Intel Core i7-5820K CPU, 32GB memory to compute all the examples. Orange circles and spheres indicate obstacles in Figs. 2 and 4. The videos corresponding to these examples can be found in the supplementary material.

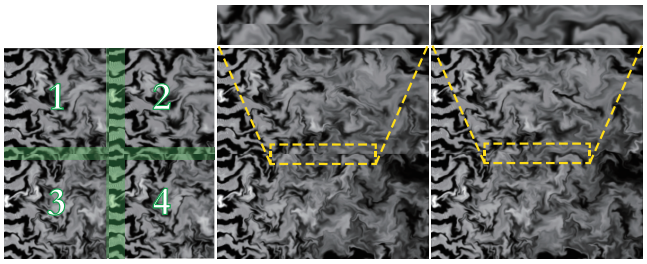
Figs. 2 and 3 show comparisons of results created by using our method and those created by importing flow, in 2D flow fields. Fig. 2 (a) and (b) are the input flow fields, (a) is rising smoke with a circular shaped obstacle, and (b) is rising smoke directed to move upward. The number of grid points is 192×256 for each flow field and the blue and red curves indicate the user specified region and the copied region, respectively, same as Fig. 1. Fig. 2 (c) is generated by importing the velocity fields from (a) to (b), and (d) is created by using our method. In Fig. 2 (c), since incompressibility is not guaranteed near the boundary of the synthesis region, the smoke densities disappear in the region within the yellow dotted curve. To confirm that the incompressibility is not guaranteed, we show the divergence of (c) in (e). The red and blue parts in the divergence fields indicate maximum and minimum value, respectively. As shown in Fig. 2 (e), the divergence around the boundary of the synthesis region is large. In contrast, for our result in Fig. 2 (d) the incompressibility is guaranteed, and, furthermore, our result can combine multiple flow fields without discontinuities. In Fig. 3, a combined flow is created by tiling the four input flow fields shown in Fig. 3 (a). The number of grid points is 256×256 for each input flow field and 512×512 for the combined flow fields. For this



(a) source field (c) importing only (e) divergence of (c)

(b) target field (d) our result (f) divergence of (d)

Figure 2: *Comparison of our method and a flow created by importing input flows. (a) and (b) are input flow fields.*

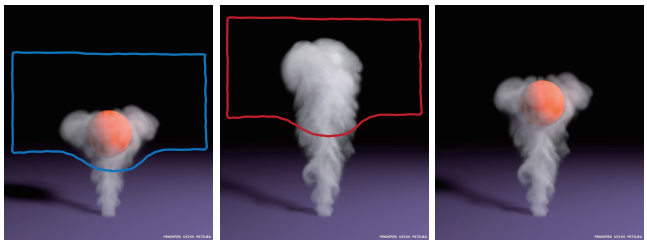


(a) input flow fields (b) inporting only (c) our result

Figure 3: *Comparison of our method and a flow created by importing input flows.*

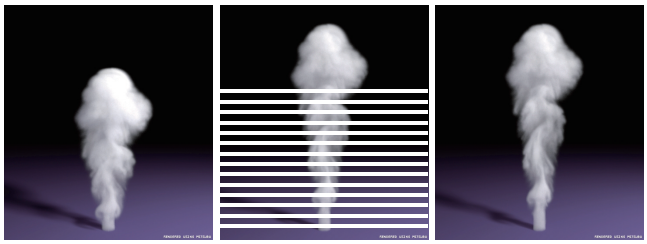
result, we interpolate the cross shaped region (marked in green in Fig. 3 (a)). Fig. 3 (b) is generated by copying the velocity fields, and (c) is synthesized by using our method. In (b), there are discontinuities within the yellow dotted curve. In contrast, our method smoothly combines the four input flow fields as shown in Fig. 3 (c). These results show that our method is effective for combining multiple flow fields. The computation times for our method are 0.2 and 0.7 seconds per frame for Fig. 2 (d) and Fig. 3 (c), respectively. On the other hand, the computation times for fluid simulation with the same number of grid points are 1.1 and 18.0 seconds per frame for Fig. 2 and Fig. 3, respectively. Since Poisson's equation is solved within a small fraction of the total area only, the desired flow can be computed faster using our method than by using fluid simulation.

Figs. 4 - 6 show results created by applying our method to 3D flow fields. Fig. 4 (a) is rising smoke with a spherical shaped obstacle, and (b) is rising smoke directed to move upward. Fig. 4 (c) is created by combining the two datasets (a) and (b). For 3D flow fields, synthesis regions are specified by a 2D curve, and the region for the depth direction is same with the size for the simulation space. The blue and red curves indicate the user specified region and the copied region, respectively, same as Fig. 1. The number of grid points is $256 \times 256 \times 256$ for each flow field. The computation time using



(a) source field (b) target field (c) our result

Figure 4: *Experimental result in 3D flow fields.*



(a) input flow field (b) synthesis regions (c) our result

Figure 5: *Partially extending a smoke animation by using our method.*

our method is about 10.0 times faster than for fluid simulation.

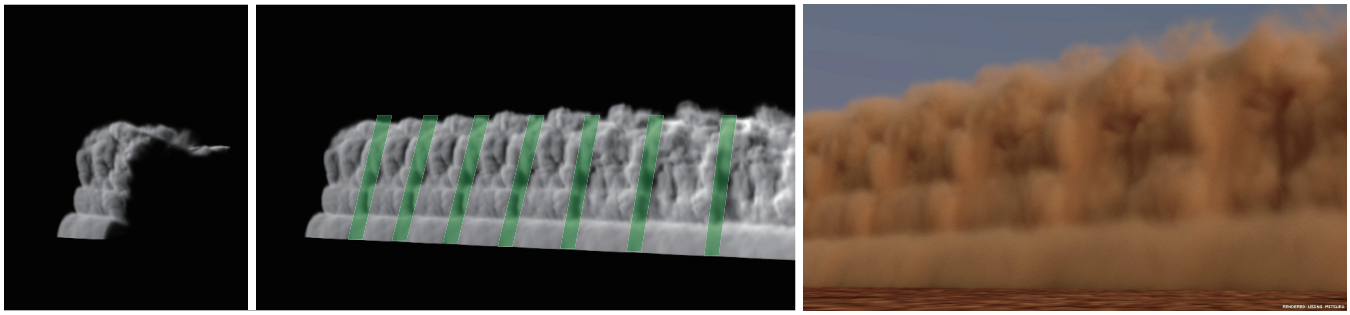
In Fig. 5, we create a flow field extended from a single input flow field. Fig. 5 (a) is the input flow field. Fig. 5 (c) is our result created by extending the lower half of (a) as shown in Fig. 5 (b). In other words, we interpolate the white regions shown in Fig. 5 (b). The number of grid points is $256 \times 256 \times 256$ for the input flow and $256 \times 256 \times 320$ for the extended flow field. The computation time using our method is about 28.0 times faster than for fluid simulation with $256 \times 256 \times 320$ grid points. Since the lower half is extended, using our method the flow field can be extended while preserving the shape of the smoke in the upper half of (a).

In Fig. 6, we create a large-scale dust storm animation from a single flow field simulated in a small-scale area. Fig. 6 (a) shows the input flow field. Fig. 6 (c) shows the result created by placing the input flow at multiple positions and interpolating the green regions as shown in Fig. 6 (b). The number of grid points is $256 \times 64 \times 128$ for the input flow field and $256 \times 456 \times 128$ for the combined flow field. The computation time using our method is about 6.5 times faster than for fluid simulation with $256 \times 456 \times 128$ grid points. As the result shows, our method can be used to efficiently create a large-scale flow field by combining multiple small-scale flow fields.

6 Conclusion and Limitations

We have proposed a method for combining existing fluid flow fields while preserving the incompressibility. Our method combines multiple flow fields by solving minimization problem. In addition, incompressible flow fields can be obtained by minimizing the divergence of the velocity field. By using our method, new desired flow fields can be efficiently and easily created by combining existing flow fields.

One of the limitations of our method is that the flow fields generated might have discontinuities at the boundaries of the synthesis regions if the input flows have large differences. To address this problem, we plan to introduce a mechanism to automatically detect



(a) input flow field (b) synthesis regions (c) our result

Figure 6: *Large-scale dust storm animation created by combining flow fields simulated in a small-scale area.*

regions with smaller error by solving a minimization problem using the user specified regions for the initial conditions. In addition, in the result of dust storm (Fig. 6), pattern of repeat of the input flow is confirmed because a single flow field is used as the input. We plan to develop a method for reducing such pattern in combined flow field.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP15H05924.

References

BOJSEN-HANSEN, M., AND WOJTAN, C. 2016. Generalized non-reflecting boundaries for fluid re-simulation. *ACM Transactions on Graphics 35*, 4, Article 96.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. AK Peters.

FEDKIW, R., STAM, J., AND JANSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH 2001*, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND ARIKAN, O. 2003. Animating suspended particle explosions. In *Proceedings of ACM SIGGRAPH 2003*, 708–715.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, 23–30.

FULLER, A. R., KRISHNAN, H., MAHROUS, K., HAMANN, B., AND JOY, K. I. 2007. Real-time procedural volumetric fire. In *Proceeding of the 2007 symposium on Interactive 3D graphics and games*, 175–180.

KIM, T., AND DELANEY, J. 2013. Subspace fluid re-simulation. *ACM Transactions on Graphics 32*, 4, Article 62.

LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of flames for a production environment. *ACM Transactions on Graphics 21*, 3, 729–735.

MIYAZAKI, R., DOBASHI, Y., AND NISHITA, T. 2002. Simulation of cumuliform clouds based on computational fluid dynamics. In *Proceedings of EUROGRAPHICS 2002 Short Presentations*, 405–410.

NGUYEN, D. Q., FEDKIW, R., AND JENSEN, H. W. 2002. Physically based modeling and animation of fire. *ACM Transactions on Graphics 21*, 3, 721–728.

PIGHIN, F., COHEN, J., AND SHAH, M. 2004. Modeling and editing flows using advected radial basis functions. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 223–232.

RAVEENDRAN, K., WOJTAN, C., THUEREY, N., AND TURK, G. 2014. Blending liquids. *ACM Transactions on Graphics 33*, 4, Article 137.

SATO, S., DOBASHI, Y., IWASAKI, K., YAMAMOTO, T., AND NISHITA, T. 2014. Deformation of 2D flow fields using stream functions. In *Proceedings of SIGGRAPH Asia 2014 Technical Briefs*, Article 4.

SATO, S., DOBASHI, Y., YUE, Y., IWASAKI, K., AND NISHITA, T. 2015. Incompressibility-preserving deformation for fluid flows using vector potentials. *The Visual Computer 31*, 6, 959–965.

STAM, J. 1999. Stable fluids. In *Proceedings of ACM SIGGRAPH 1999, Annual Conference Series*, 121–128.

TREUILLE, A., LEWIS, A., AND POPOVIC, Z. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics 25*, 3, 826–834.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Transaction on Graphics 28*, 3, Article 39.

YNGVE, G. D., O'BRIEN, J. F., AND HODGINS, J. K. 2000. Animating explosions. In *Proceedings of ACM SIGGRAPH 2000*, 29–36.