CGI2015 manuscript No.
(will be inserted by the editor)

Incompressibility-Preserving Deformation for Fluid Flows Using Vector Potentials

Syuhei Sato · Yoshinori Dobashi · Yonghao Yue · Kei Iwasaki · Tomoyuki Nishita

Abstract Physically-based fluid simulations usually require expensive computation cost for creating realistic animations. We present a technique that allows the user to create various fluid animations from an input fluid animation sequence, without the need for repeatedly performing simulations. Our system allows the user to deform the flow field in order to edit the overall fluid behavior. In order to maintain plausible physical behavior, we ensure the incompressibility to guarantee the mass conservation. We use a vector potential for representing the flow fields to realize such incompressibility-preserving deformations. Our method first computes (time-varying) vector potentials from the input velocity field sequence. Then, the user deforms the vector potential, and the system computes the deformed velocity field by taking the curl operator on the vector potential. The incompressibility is thus obtained by construction. We show various examples to demonstrate the usefulness of our method.

Keywords flow deformation · incompressibility · vector potential

Syuhei Sato
UEI Research, Tokyo, Japan
E-mail: syuhei.sato@uei.co.jp

Yoshinori Dobashi
Hokkaido University, JST CREST, UEI Research, Sapporo, Japan

Yonghao Yue
Columbia University, NY, USA

Kei Iwasaki
Wakayama University, UEI Research, Wakayama, Japan

Tomoyuki Nishita
UEI Research, Hiroshima Shudo University, Tokyo, Japan

1 Introduction

Physically-based simulations are ubiquitous for creating realistic fluid animations. Their expensive computation cost is, however, usually a common drawback. Because a number of simulations with different parameters have to be tested before the desired animations are obtained, the total simulation time could be too long.

In the production environment, artists often create various fluid animations by deforming the simulated flow fields or density fields. For example, one may use procedural methods to deform the flow fields [8,14], or may directly deform the density fields. Although these approaches allow for generating various stylized animations without the need for repeatedly running the simulations, the results are typically not guaranteed to satisfy the physical conservation laws; violating the mass conservation can easily lead to noticeable visible artifacts such as increasing/decreasing fluid mass.

We present a method for deforming existing fluid flow fields while preserving incompressibility (i.e., mass conservation), by using vector potentials. Our method first computes a (time-varying) vector potential $\mathbf{\Psi}$ from the input velocity field sequence. Then, the user deforms the vector potential, and the system computes the deformed velocity field by taking its curl. Because we have the identity $\nabla \cdot \nabla \times \mathbf{\Psi} = 0$ for any vector potential $\mathbf{\Psi}$, our technique guarantees the incompressibility by construction. By using our method, the user can deform existing flow fields to efficiently create various fluid animations.

2 Related Work

Solving the Navier-Stokes equations. Since Stam had introduced an unconditionally stable solver [21] for

the Navier-Stokes equations, many methods have been proposed for simulating various fluid phenomena [2,5–7,10,15]. Usually, parameters need to be adjusted to obtain a desired animation; a naïve adjustment would require repeatedly performing simulations with different parameter sets, possibly leading to an unacceptable workload.

Fluid control. Several control methods have been presented for creating fluid animations with the desired shapes (e.g. characters, logos) [4,20,24], or for enabling the fluids to move along user specified 3D curves (paths) [13]. These methods control fluid motions by adding external forces to guide the fluids. Using these methods, the user needs to typically wait for multiple simulations to finish, in order to create various fluid animations. Our method offers an alternative way to create various fluid animations, without the need for re-running any simulation.

Model reduction. Model reduction methods [23,25] prepare many sets of velocity fields obtained by simulating fluids with various parameters and initial conditions. Then, principal component analysis (PCA) is applied to the velocity fields to generate basis functions. At the cost of expensive precomputation, the flow field can be updated efficiently by computing the Navier-Stokes equations in this basis space. Kim et al. [12] presented a method for efficient re-simulation with different parameter settings, by applying PCA to a single set of velocity fields to generate the basis functions. In these methods, the user is limited to creating flow fields that are represented by a linear combination of the basis functions. In addition, memory consumption for solving PCA can be a bottleneck for a large database.

Procedural methods. Procedural methods can be used to generate various flows with relatively low computation cost. For example, for editing fire animations, methods (e.g., [8,14]) have been presented to enable the user to deform the fire via a curve representing its route. Pighin et al. [17] proposed a method for editing simulated flow fields via advected radial basis functions. Since these procedural methods typically do not take into account physical constraints like the incompressibility, unrealistic results can be produced.

Flow deformation. By using scalar stream functions, we have previously shown that 2D flow fields can be deformed while guaranteeing the incompressibility [18]. In this paper, we show that 3D flow fields can be deformed in an incompressible way as well, by using vector potentials.

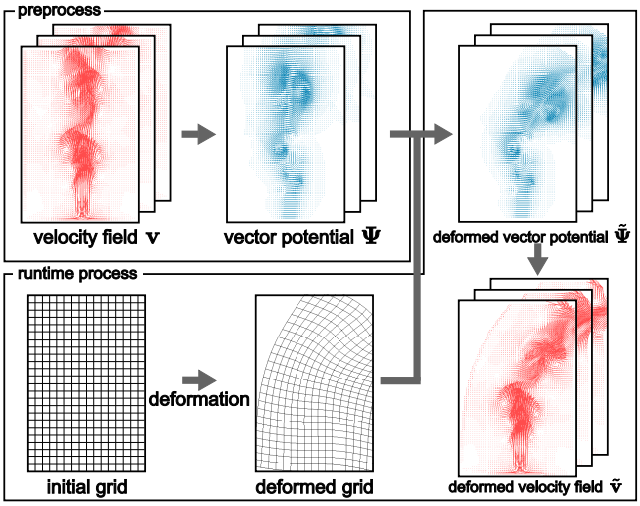

Fig. 1 Overview of our method. For the sake of visual clarity, each 3D vector field is illustrated as a 2D vector field.

Computing vector potentials. The computation of vector potentials from velocity fields appears in computational fluid dynamics as well. For example, in the vortex methods (e.g., [3,9,16,26,27]), one needs to convert the vorticity field $\omega = \nabla \times \mathbf{v}$ back to the velocity field $\mathbf{v}$, which can be regarded as the vector potential of ω. To our knowledge, our method is the first that uses vector potentials for fluid editing. We discuss boundary conditions and equations that are appropriate for our problem.

3 Input and Assumptions

Our input is a set of incompressible velocity fields $\mathbf{v}(t)$ (i.e., $\nabla \cdot \mathbf{v}(t) = 0$), where $t = 0, 1, \cdots, T-1$ represents the frame count, and T is the number of frames of the input velocity field. We assume that $\mathbf{v}(t)$ is sufficiently smooth and is defined in a simply connected closed domain $\Omega \subset \mathbb{R}^3$ (i.e., Ω is genus 0). In addition, we can naturally assume $\mathbf{v}(t)$ is tangential to $\partial\Omega$:

$$(\mathbf{v}(t)) \cdot \mathbf{n} = 0 \qquad \text{at the boundary } \partial\Omega, \tag{1}$$

where $\mathbf{n}$ is normal to the boundary.

In addition, we assume the fluid, represented as a density field, is advected along $\mathbf{v}(t)$. Our method generates various animations by replacing the velocity field, and then advecting the fluid along the new velocity field. Henceforth, we omit the notation t for brevity.

4 Overview

Instead of manipulating the velocity or density fields directly, we deal with the vector potential to guarantee

the incompressibility of the deformed flow. In a simple connected closed domain, the Helmholtz-Hodge decomposition theory [1,22] states that a sufficiently smooth vector field $\mathbf{v}$ can be decomposed as

$$\mathbf{v} = \nabla\times\boldsymbol{\Psi} + \nabla p, \tag{2}$$

where $\boldsymbol{\Psi}$ is a vector potential, p is a scalar field. $\nabla\times\boldsymbol{\Psi}$ is tangential to the boundary of the domain:

$$(\nabla\times\boldsymbol{\Psi})\cdot\mathbf{n} = 0 \qquad \text{at the boundary,} \tag{3}$$

and ∇p is perpendicular to the boundary.

Since our input velocity field $\mathbf{v}$ is incompressible, we have $\nabla p = 0$ (as in Appendix A), and hence

$$\mathbf{v} = \nabla\times\boldsymbol{\Psi}. \tag{4}$$

Because the identity $\nabla\cdot(\nabla\times\boldsymbol{\Psi}) = 0$ is satisfied for an arbitrary vector field $\boldsymbol{\Psi}$, the curl of the vector potential is always incompressible. Therefore, if we manipulate the fluid via the vector potential, and then convert the vector potential back to the velocity field, we can guarantee the incompressibility of the deformed flow by construction.

Fig. 1 shows an overview of our method. In the preprocess, the velocity field $\mathbf{v}$ at each time step is converted into the vector potential $\boldsymbol{\Psi}$. At runtime, the user deforms the spatial domain to obtain the deformed vector potentials $\tilde{\boldsymbol{\Psi}}$. Then, the curl operator is applied to $\tilde{\boldsymbol{\Psi}}$ to generate the deformed velocity field $\tilde{\mathbf{v}}$. Finally, the density fields are advected along $\tilde{\mathbf{v}}$.

5 Computing Vector Potentials

Although the input incompressible vector field $\mathbf{v}$ can be related to a vector potential $\boldsymbol{\Psi}$ via $\mathbf{v} = \nabla\times\boldsymbol{\Psi}$, as in Eq.(4), $\boldsymbol{\Psi}$ is usually not unique. Suppose that $\boldsymbol{\Psi}$ satisfies $\mathbf{v} = \nabla\times\boldsymbol{\Psi}$, then any vector potential $\boldsymbol{\Psi}' = \boldsymbol{\Psi} + \nabla q$ also satisfies $\mathbf{v} = \nabla\times\boldsymbol{\Psi}'$ because $\nabla\times\nabla q = 0$ for an arbitrary scalar field q. We discuss how we pin down this freedom in a way suitable for deforming a time-varying vector field.

First, we observe that according to Eq.(2), the vector potential itself can be decomposed as $\boldsymbol{\Psi} = \nabla\times\boldsymbol{\Phi} + \nabla s$, where $\boldsymbol{\Phi}$ is a vector field and s is a scalar field. Now, suppose that $\boldsymbol{\Psi}$ is deformed via a map F, which, for example, maps each position $\mathbf{X}$ to a new position $\mathbf{x}$. Under this deformation, a vector valued function $\mathbf{g}(\mathbf{X})$ will be transformed to $\tilde{\mathbf{g}}(\mathbf{x}) = \mathbf{g}(\mathbf{X})$[1], which we write $F(\mathbf{g}) = \tilde{\mathbf{g}}$. For vector valued functions

[1] Or, we could consider $\tilde{\mathbf{g}}(\mathbf{x}) = \frac{\partial\mathbf{x}}{\partial\mathbf{X}}\mathbf{g}(\mathbf{X})$ if we want to apply local rotation, stretch and shear to reorient the vector valued function.

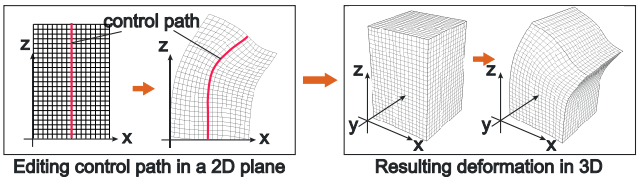

Fig. 2 Deformation using a control path.

$\mathbf{g}$ and $\mathbf{h}$, we have $F(\mathbf{g}+\mathbf{h}) = \tilde{\mathbf{g}} + \tilde{\mathbf{h}} = F(\mathbf{g}) + F(\mathbf{h})$. Then, the deformed vector potential will have the form $F(\boldsymbol{\Psi}) = F(\nabla\times\boldsymbol{\Phi} + \nabla s) = F(\nabla\times\boldsymbol{\Phi}) + F(\nabla s)$. For an arbitrary deformation, $\nabla\times F(\nabla s)$ is generally non-zero, because $F(\nabla s)$ might no longer be a curl free field. We wish to pin down ∇s such that $\nabla\times F(\nabla s)$ is coherent in time so that it does not cause a fluctuation in the resulting velocity field. Specifically, we enforce $\nabla s = \mathbf{0}$, by setting

$$\boldsymbol{\Psi}\cdot\mathbf{n} = 0 \qquad \text{at the boundary, and} \tag{5}$$
$$\nabla\cdot\boldsymbol{\Psi} = 0 \qquad \text{for the entire domain.} \tag{6}$$

To compute the desired vector potential $\boldsymbol{\Psi}$ from the velocity field $\mathbf{v}$, we first apply $\nabla\times$ to the both sides of Eq.(4) and obtain

$$\nabla\times(\nabla\times\boldsymbol{\Psi}) = \nabla\times\mathbf{v}. \tag{7}$$

Under our setting, the solution to Eq.(7) satisfies Eq.(4) as in Appendix B. Then, from the identity

$$\nabla\times(\nabla\times\boldsymbol{\Psi}) = \nabla(\nabla\cdot\boldsymbol{\Psi}) - \nabla^2\boldsymbol{\Psi}, \tag{8}$$

we have

$$\nabla(\nabla\cdot\boldsymbol{\Psi}) - \nabla^2\boldsymbol{\Psi} = \nabla\times\mathbf{v}. \tag{9}$$

Finally, substituting Eq.(6) into Eq.(9), we obtain

$$-\nabla^2\boldsymbol{\Psi} = \nabla\times\mathbf{v}. \tag{10}$$

We solve this Poisson equation with the boundary conditions $\boldsymbol{\Psi}\cdot\mathbf{n} = 0$ (5), $(\nabla\times\boldsymbol{\Psi})\cdot\mathbf{n} = 0$ (3) and $\nabla\cdot\boldsymbol{\Psi} = 0$ (6) to obtain the vector potential $\boldsymbol{\Psi}$. As in Appendix C, we show that Eq.(10) is equivalent to Eq.(7) under our setting. Hence the solution to Eq.(10) together with the boundary conditions satisfies Eq.(7) and in turn Eq.(4). To obtain numerical solution to Eq.(10), we used the biconjugate gradient stabilized method (BiCGSTAB).

6 Generation of Deformed Velocity Field

In our method, we assume the input velocity fields are given as a sequence of voxel data. We represent the vector potentials using a grid as well. The deformation is applied to the grid directly. We can in general use any

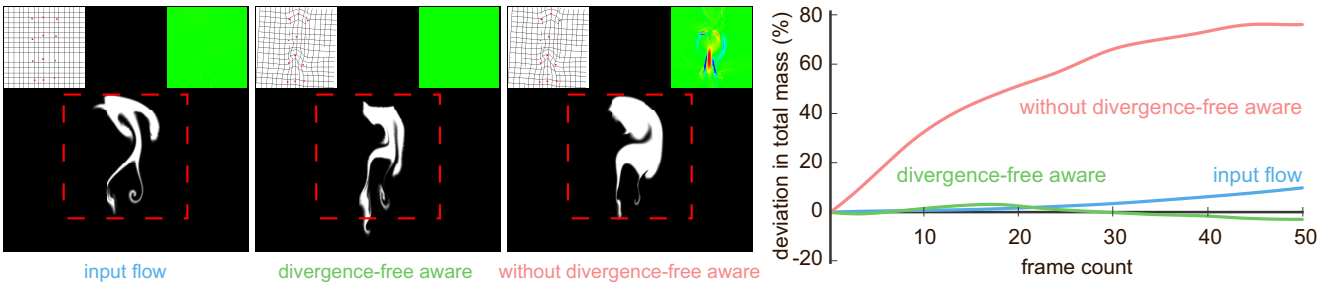

Fig. 3 The importance of divergence-free aware deformation. The top left insets show the underlying grid of the domain. The top right insets show the divergence of the velocity field in the region marked by the red dashed rectangular: blue, green, red colors indicate negative, zero, positive divergence, respectively. The deviation of the total mass is computed as $(m_t/m_0 - 1)$, where the total mass m is computed by integrating the density over the entire simulation domain.

deformation method developed for images or meshes for this purpose. However, if deformations that cause fold-overs are applied, large velocities might be generated because the fold-overs cause discontinuities in the vector fields. In addition, unintentional drastic changes in the velocity values might occur if the degrees of deformations are too large, because the deformed grid in the largely deformed region could be under resolved.

To compute the new deformed vector potential $\tilde{\mathbf{\Psi}}$, defined in the world coordinates, we first resample $\mathbf{\Psi}$ stored in the deformed grid to the cartesian grid storing $\tilde{\mathbf{\Psi}}$, and then apply the local rotation corresponding to the deformation, in order to get the correct orientations for the vector potentials. Next, the deformed velocity field $\tilde{\mathbf{v}}$ is obtained by applying the curl operator to $\tilde{\mathbf{\Psi}}$: $\tilde{\mathbf{v}} = \nabla\times\tilde{\mathbf{\Psi}}$. The resulting velocity field always satisfies the divergence-free condition since $\nabla\cdot(\nabla\times\tilde{\mathbf{\Psi}}) = 0$.

7 Results

We used a desktop PC with an Intel Core i7 3930K CPU, 32GB memory to compute all the examples. To render 3D results shown in Figs. 5, 6 and the right image in Fig. 7, we used the physically-based renderer "Mitsuba" [11]. We used control-points or control-paths to specify the deformation in our system (Fig. 2 shows an example of using a control-path). The grid is deformed according to the control-points or control-paths using a method based on moving least squares [19]. In addition, the deformation is executed on a 2D plane, and this deformation is then applied to each plane perpendicular to the y direction of the 3D grid (Fig. 2). Videos corresponding to the following examples can be found in the supplementary material (Online Resource 1).

The importance of divergence-free aware deformation. The ability to preserve the incompressibility of the velocity fields is important to conserve the total mass. We see this importance in Fig. 3. For the sake of visual clarity, we used a 2D flow field as demonstration. The 2D analog of the 3D vector potential is a scalar function called stream function, computed from the 2D velocity field using our previous work [18].

In this example, a non-zero density field is advected along a time-varying velocity field generated using a 2D fluid simulation, where constant upward velocities are set at the center bottom of the simulation space to drive the flow. No density is added or removed at runtime. If we use the moving least squares to deform the velocity field directly without using the stream function (denoted as "without divergence-free aware"), the resulting deformed velocity field will have non-zero divergence, which can in turn cause the total mass to deviate largely from its initial value. In contrast, by working with the potential, it is possible to guarantee the incompressibility all the time, which significantly reduces the mass deviation. We observed the same tendency in deforming 3D flow fields.

Comparing the edited results against simulated results in 3D. Fig. 4 shows a comparison of results created by our method and fluid simulations. Fig. 4(a) shows the original input smoke animation. Figs. 4(b') through (d') are created by the fluid simulations, where uniform external forces are applied at the left boundary of the simulation space to make the smoke flow to the right. Figs. 4(b) through (d) are the deformed results obtained using our method, showing that we can mimic such flows by using our deformation, and the results are fairly close under small deformations. Moreover, we see that the deformed animation is still plausible for a relatively large deformation in this setting.

3D fluid editing. Next, we show applications of our flow deformation to 3D fluid editing. The timing information is summarized in Table 1. The computation time of our method at runtime is about 50.0 times faster than the simulation, while the precomputation is only

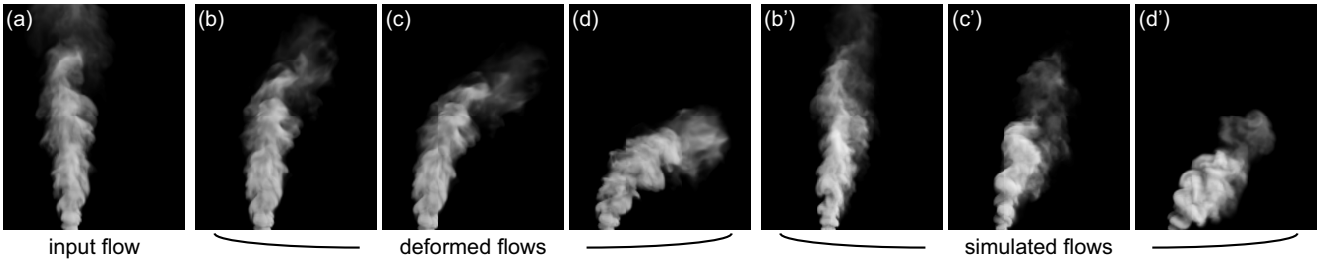

Fig. 4 Comparison of results generated by our method and a fluid simulation.

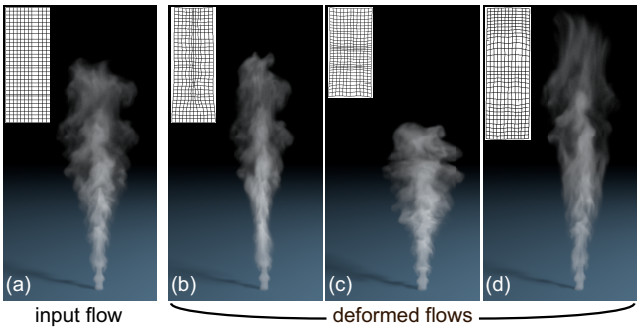

Fig. 5 Various stylized animations (b) to (d) from a single input flow (a). The insets show the underlying grid of the domain.

Fig. 6 Examples of smoke animations rising from multiple chimneys using the results from Figs. 4(b) through (d).

Table 1 Computation times per frame measured in seconds. T_{ns} is for the fluid simulation. T_p and T_r are for preprocess and runtime in our method, respectively.

Fig.	grid resolution	T_{ns}	T_p	T_r
4	$256 \times 128 \times 384$	146	298	2.9
5, 7	$192 \times 192 \times 512$	234	407	4.4

2.0 times longer than the simulation for generating the input flow. Hence the trial-and-error process for creating various animations with our method is much more efficient than re-running simulations.

From a single input flow (Fig. 5(a)), we can create stylized animations, namely "slender" (Fig. 5(b)), "fat" (Fig. 5(c)) or "tall" smokes (Fig. 5(d)), by squashing the grid horizontally, vertically, or by vertically stretching the grid, respectively.

In Fig. 6, we created a scene with rising smokes from multiple chimneys, by using the results from Figs. 4(b) through (d). With our method, we can efficiently create synthetic scenes with multiple varied fluid animations from a single simulated dataset.

Fig. 7 shows a stylized smoke from a magic lamp. The input flow is a vertically rising smoke (Fig. 7(a)), generated by using a fluid simulation. The source of the smoke is located at the center-bottom of the simulation space. By deforming the flow using our method, we can create a swirling smoke motion as in Fig. 7(b), which can be in turn used to generate the image on the right in Fig. 7, for an interesting smoke motion.

Limitations. Since we deform the vector potential instead of the velocity field, the resulting deformation in the velocity field can be non-intuitive. For example, because the $x-$, $y-$ and $z-$ components of the vector potential are coupled through the expression $\mathbf{v} = \nabla \times \mathbf{\Psi}$, deforming the vector potential in a plane will usually also result in a change in the velocity field in the direction perpendicular to the plane. In addition, too large deformation can result in a non-intuitive flow as well. We wish to quantify the satisfaction of the deformation with respect to the degree and type of the deformation in the future.

8 Conclusions and Future Work

We have presented a method for deforming fluid flow fields while preserving the incompressibility. The divergence free condition is satisfied by construction: by working with the vector potentials and converting them back to the velocity fields, we can automatically obtain the incompressibility. In addition, we have presented a method for pinning down the freedom in the vector

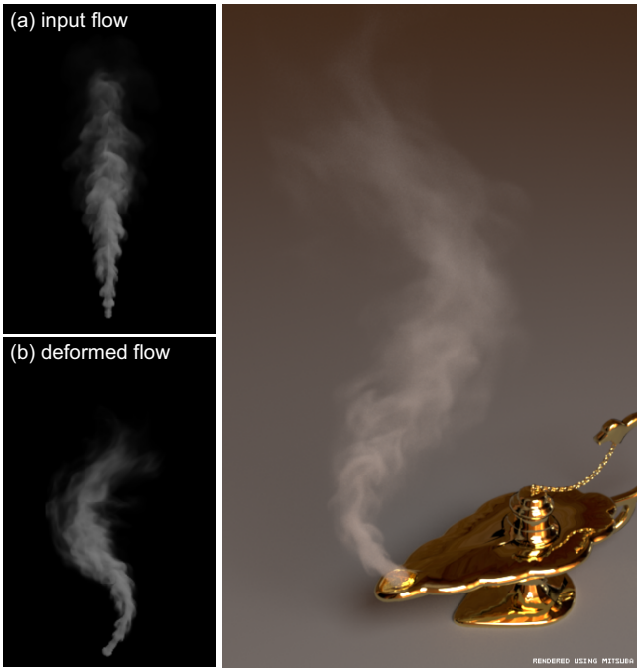

Fig. 7 Example of a smoke animation rising from a magic lamp.

potentials, such that it does not contain any curl free component.

As future work, we wish to identify an intuitive range of deformation. In addition, we are planning to develop a method that can deform fluid flow fields while conserving the momentum as well.

Acknowledgements

This work was supported in part by the JSPS Postdoctoral Fellowships for Research Abroad.

References

1. Bhatia, H., Norgard, G., Pascucci, V., Bremer, P.: The Helmholtz-Hodge decomposition–a survey. IEEE Transactions on Visualization and Computer Graphics **19**(8), 1386–1404 (2013)
2. Bridson, R.: Fluid Simulation for Computer Graphics. AK Peters (2008)
3. Cottet, G.H., Koumoutsakos, P.D.: Vortex Methods: Theory and Practice. Cambridge University Press (2000)
4. Fattal, R., Lischinski, D.: Target-driven smoke animation. ACM Transactions on Graphics **23**(3), 439–446 (2004)
5. Fedkiw, R., Stam, J., Jansen, H.W.: Visual simulation of smoke. In: Proceedings of ACM SIGGRAPH 2001, pp. 15–22 (2001)
6. Feldman, B.E., O'Brien, J.F., Arikan, O.: Animating suspended particle explosions. In: Proceedings of ACM SIGGRAPH 2003, pp. 708–715 (2003)
7. Foster, N., Fedkiw, R.: Practical animation of liquids. In: Proceedings of ACM SIGGRAPH 2001, pp. 23–30 (2001)
8. Fuller, A.R., Krishnan, H., Mahrous, K., Hamann, B., Joy, K.I.: Real-time procedural volumetric fire. In: Proceeding of the 2007 symposium on Interactive 3D graphics and games, pp. 175–180 (2007)
9. Gamito, M.N., Lopes, P.F., Gomes, M.R.: Two-dimensional simulation of gaseous phenomena using vortex particles. In: In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation, pp. 3–15. Springer-Verlag (1995)
10. Hong, W., House, D.H., Keyser, J.: Adaptive particles for incompressible fluid simulation. The Visual Computer **24**(7-9), 535–543 (2008)
11. Jakob, W.: Mitsuba renderer (2010). Http://www.mitsuba-renderer.org
12. Kim, T., Delaney, J.: Subspace fluid re-simulation. ACM Transactions on Graphics **32**(4), Article 62 (2013)
13. Kim, Y., Machiraju, R., Thompson, D.: Path-based control of smoke simulations. In: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 33–42 (2006)
14. Lamorlette, A., Foster, N.: Structural modeling of flames for a production environment. ACM Transactions on Graphics **21**(3), 729–735 (2002)
15. Nguyen, D.Q., Fedkiw, R., Jensen, H.W.: Physically based modeling and animation of fire. ACM Transactions on Graphics **21**(3), 721–728 (2002)
16. Park, S.I., Kim, M.J.: Vortex fluid for gaseous phenomena. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 261–270 (2005)
17. Pighin, F., Cohen, J., Shah, M.: Modeling and editing flows using advected radial basis functions. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 223–232 (2004)
18. Sato, S., Dobashi, Y., Iwasaki, K., Yamamoto, T., Nishita, T.: Deformation of 2D flow fields using stream functions. In: Proceedings of SIGGRAPH Asia 2014 Technical Briefs, Article 4 (2014)
19. Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM Transactions on Graphics **25**(3), 533–540 (2006)
20. Shi, L., Yu, Y.: Taming liquids for rapidly changing targets. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 229–236 (2005)
21. Stam, J.: Stable fluids. In: Proceedings of ACM SIGGRAPH 1999, Annual Conference Series, pp. 121–128 (1999)
22. Tong, Y., Lombeyda, S., Hirani, A.N., Desbrun, M.: Discrete multiscale vector field decomposition. ACM Transactions on Graphics **22**(3), 445–452 (2003)
23. Treuille, A., Lewis, A., Popovic, Z.: Model reduction for real-time fluids. ACM Transactions on Graphics **25**(3), 826–834 (2006)
24. Treuille, A., McNamara, A., Popovic, Z., Stam, J.: Keyframe control of smoke simulations. ACM Transactions on Graphics **22**(3), 716–723 (2003)
25. Wicke, M., Stanton, M., Treuille, A.: Modular bases for fluid dynamics. ACM Transactions on Graphics **28**(3), Article 39 (2009)
26. Yaeger, L., Upson, C., Myers, R.: Combining physical and visual simulation-creation of the planet Jupiter for the film "2010". In: Proceedings of ACM SIGGRAPH '86, pp. 85–93 (1986)

27. Zhang, X., Bridson, R.: A PPPM fast summation method for fluids and beyond. ACM Trans. Graph. **33**(6), 206:1–206:11 (2014). DOI 10.1145/2661229.2661261. URL http://doi.acm.org/10.1145/2661229.2661261

A $\nabla p = 0$ for an Incompressible Vector Field v

Taking the divergence of both sides of Eq.(2), we have $0 = \nabla \cdot \mathbf{v} = \nabla^2 p$. Because $\mathbf{v}$ and $\nabla \times \boldsymbol{\Psi}$ are both tangential to the boundary, we have $(\nabla p) \cdot \mathbf{n} = 0$ at the boundary. Therefore, the solution to p is $p = const.$, hence $\nabla p = \mathbf{0}$.

B Equivalence Between Eq.(4) and Eq.(7)

Eq.(7) can be rewritten as $\nabla \times (\nabla \times \boldsymbol{\Psi} - \mathbf{v}) = 0$. Let $\mathbf{A} = \nabla \times \boldsymbol{\Psi} - \mathbf{v}$. Since $\nabla \cdot \nabla \times \boldsymbol{\Psi} = \nabla \cdot \mathbf{v} = 0$, we have $\nabla \cdot \mathbf{A} = 0$. In addition, since $(\nabla \times \boldsymbol{\Psi}) \cdot \mathbf{n} = 0$ and $\mathbf{v} \cdot \mathbf{n} = 0$, we have $\mathbf{A} \cdot \mathbf{n} = 0$. From $\nabla \times \mathbf{A} = \mathbf{0}$, $\nabla \cdot \mathbf{A} = 0$, and $\mathbf{A} \cdot \mathbf{n} = 0$, we have $\mathbf{A} = \mathbf{0}$, which gives Eq.(4).

C Equivalence Between Eq.(7) and Eq.(10)

First, we show $\nabla \cdot \boldsymbol{\Psi} = 0$ given Eq.(10) and the corresponding boundary conditions. From Eq.(10), we have $\nabla \cdot (-\nabla^2 \boldsymbol{\Psi}) = \nabla \cdot \nabla \times \mathbf{v} = 0$. Since $\boldsymbol{\Psi}$ is sufficiently smooth, $\nabla \cdot (-\nabla^2 \boldsymbol{\Psi}) = -\nabla^2 (\nabla \cdot \boldsymbol{\Psi})$. Hence we have $\nabla^2 (\nabla \cdot \boldsymbol{\Psi}) = 0$. Writing $G = \nabla \cdot \boldsymbol{\Psi}$, we have $\nabla^2 G = 0$. Since $\nabla \cdot \boldsymbol{\Psi} = 0$ at the boundary, we have $G = 0$ at the boundary. Therefore, the solution to G is $G = 0$ for the entire domain, which yields $\nabla \cdot \boldsymbol{\Psi} = 0$. Next, from $\nabla \cdot \boldsymbol{\Psi} = 0$, Eq.(8) and Eq.(10), we obtain Eq.(7).