Deformation of 2D Flow Fields Using Stream Functions

Syuhei Sato*
UEI Research

Yoshinori Dobashi
Hokkaido University / JST CREST

Kei Iwasaki
Wakayama University / UEI Research

Tsuyoshi Yamamoto
Hokkaido University

Tomoyuki Nishita
UEI Research / Hiroshima Shudo University

Abstract

Recently, visual simulation of fluids has become an important element in many applications, such as movies and computer games. These fluid animations are usually created by physically-based fluid simulation. However, the simulation often requires very expensive computational cost for creating realistic fluid animations. Therefore, when the user tries to create various fluid animations, he or she must execute fluid simulation repeatedly, which requires a prohibitive computational time. To address this problem, this paper proposes a method for deforming velocity fields of fluids while preserving the divergence-free condition. In this paper, we focus on grid-based 2D fluid simulations. Our system allows the user to interactively create various fluid animations from a single set of velocity fields generated by the fluid simulation. In a preprocess, our method converts the input velocity fields into scalar fields representing the stream functions. At run-time, the user deforms the grid representing the scalar stream functions and the deformed velocity fields are then obtained by applying a curl operator to the deformed scalar stream functions. The velocity fields obtained by this process naturally perseveres the divergence-free condition. For the deformation of the grid, we use a method based on Moving Least Squares. The usefulness of our method is demonstrated by several examples.

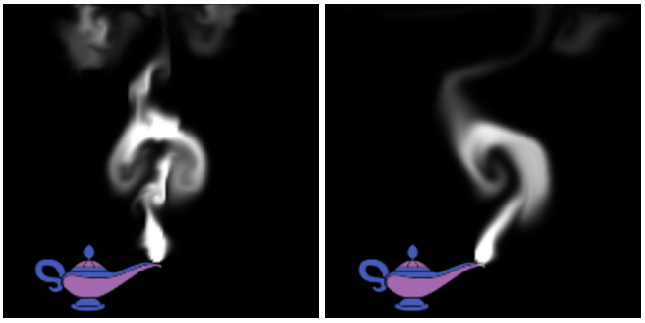
CR Categories: I.3.7 [Computer Graphics]: Animation; I.3.6 [Computer Graphics]: Methodology and Techniques;

Keywords: fluid simulation, flow field deformation, stream function, moving least squares

1 Introduction

Visual simulation of natural phenomena has become one of the most important research topics in computer graphics. Many methods have been proposed for simulating various phenomena, such as smoke, water, fire, etc [Stam 1999; Bridson 2008]. Most of the recent methods are based on computational fluid dynamics to create realistic animations and these are used in many applications such as movies and computer games to enhance the realism of synthetic scenes. However, one of the problems is the expensive computational cost of the physically-based fluid simulations. The expensive computational cost makes it difficult to create desired animations of fluids, since many simulation parameters have to be adjusted by executing the fluid simulation repeatedly.

Model reduction methods for fluid simulations have been proposed to address this problem [Treuille et al. 2006; Wicke et al. 2009]. These methods accelerate runtime computation by using precomputed database of fluid velocity fields. However, the user cannot create flow fields other than those that can be represented by a linear combination of precomputed database. In order to create various flow fields, huge data of velocity fields are needed. In addition, precomputation time for creating the database is extremely expensive. Based on the model reduction methods, a method for re-simulating fluids has been proposed [Kim and Delaney 2013]. This method can re-simulate flow fields very efficiently even when the simulation parameters are modified. However, this method uses only a single set of velocity fields in constructing the precomputed database. Therefore, the method cannot generate a result that is completely different from the original simulated data used for constructing the database. Furthermore, even when using these fast simulation method, it is still difficult to create desired fluid flow.



(a) original flow (b) deformed flow

Figure 1: *Example created by our method.*

In the production environment, artists often create desired fluid animations by deforming the simulated flow fields. Some procedural methods have also been proposed for creating various animations by deforming the flow fields [Lamorlette and Foster 2002; Fuller et al. 2007]. Although these methods can create desired animations efficiently, the resulting flow field is not physically-correct, making the animations less realistic than those created using physically-based simulation.

In this paper, we propose a method for deforming fluid flow fields in a physically-plausible way. Our method preserves the important physical law for the fluid flow, i.e., the divergence-free condition. In our method, the divergence-free condition is always satisfied even when the significant deformation is applied. The key concept behind our method is to convert fluid velocity fields into a scalar stream function. The deformation operator is applied to the stream functions and the deformed velocity fields are obtained by applying a curl operator to the deformed stream function. The resultant velocity fields naturally satisfies the divergence-free condition. Our method uses a deformation method based on Moving Least Squares [Schaefer et al. 2006]. Although we apply our deformation method to 2D smoke simulations in this paper, we demonstrate its usefulness by showing several examples. In this paper, our method

*e-mail:syuhei.sato@uei.co.jp

focuses on grid-based fluid simulation.

2 Related Work

Stam [1999] addressed the stability problem in solving the Navier-Stokes equations and made fluid simulation practical. Following this work, many methods were proposed for simulating various fluid phenomena. The details of these simulation methods are summarized in [Bridson 2008]. One of the problems with fluid simulation, however, is the expensive computational cost. Creating desired fluid animations by using these simulation methods is a time-consuming task.

In order to address this problem, model reduction methods for fluid simulations have been proposed [Treuille et al. 2006; Wicke et al. 2009]. These methods prepare many set of velocity fields obtained by simulating fluids with various parameters and initial conditions. Principal Component Analysis (PCA) are applied to these velocity field data, and obtained principal components are used as basis functions. By calculating Navier-Stokes equations in the basis space, fluid flow is simulated very efficiently. A re-simulation method based on the model reduction approach has also been developed [Kim and Delaney 2013]. This method calculates basis functions by applying PCA to a single set of velocity fields generated by fluid simulation. Flow fields with a different parameter setting can be re-simulated very efficiently by computing Navier-Stokes equations in the basis space. However, since these methods create basis functions using PCA, the user cannot create flow fields other than those represented by a linear combination of precomputed database.

Procedural methods can generate desired flows with relatively low computational cost. For example, curve-based methods for creating various fire animations have been proposed [Lamorlette and Foster 2002; Fuller et al. 2007]. These methods generate user-designed fire animations by deforming a curve representing the route of the fire. However, since these procedural methods do not take into account physical accuracy, unrealistic results could be produced. The incompressibility of the flow is not assured, too.

3 Fluid Simulation

In this paper, we use incompressible Navier-Stokes equations for simulating fluids. Motions of fluids are calculated by solving the following Navier-Stokes equations.

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v}\cdot\nabla)\mathbf{v} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{v} + \mathbf{f}, \tag{1}$$

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

where $\mathbf{v}$ is velocity, ρ is density, p is pressure, ν is kinematic viscosity coefficient, and $\mathbf{f}$ is external forces such as gravity and wind. ∇ is a gradient operator, $\nabla\cdot$ is a divergence operator, and ∇^2 is a Laplacian operator. Eq.(1) represents temporal evolution of the velocity field. Each term on the right-hand side from left to right are called advection, pressure, diffusion, and external force terms, respectively. This equation is very important for creating plausible flow fields. When we simulate incompressible fluids, Eq.(2) has to be considered.

Smoke is simulated by using the velocity field calculated by advecting smoke densities according to the velocity field obtained by solving Navier-Stokes equations, that is,

$$\frac{\partial D}{\partial t} = -(\mathbf{v}\cdot\nabla)D + D_s, \tag{3}$$

where D is smoke density, D_s is the density added from smoke sources.

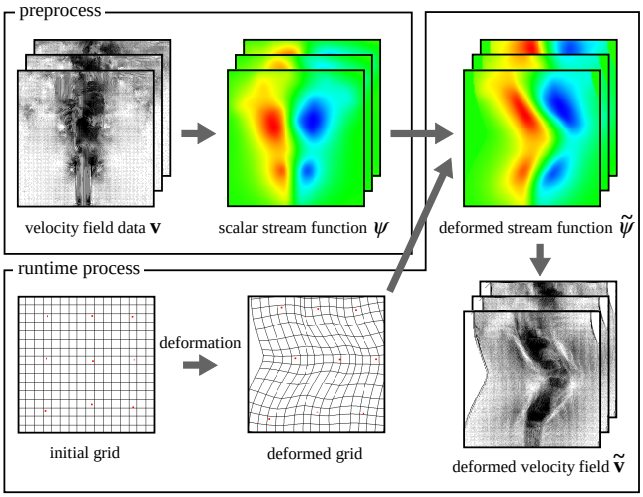


Figure 2: *Overview of our method.*

The velocity field used as input to our method is obtained by solving Eqs.(1) and (2) . Our method deforms the velocity field and smoke densities are advected according to the deformed velocity field.

4 Deformation of Flow Field

This section describes our method for deforming a flow field. Fig. 2 shows an overview of our method. Our method consists of two processes: a preprocess and a run-time process. Firstly, in the preprocess, the dynamic velocity field $\mathbf{v}$ is generated by using the fluid simulation described in the previous section. Then, the velocity field at each time step, $\mathbf{v}(t)$, is converted into a scalar stream function $\psi(t)$, where $t(=0,1,\cdots,T-1)$ represents a frame number, and T is a number of frames of the prepared dynamic velocity field. Next, at the runtime process, a deformation grid for deforming the stream function is prepared. Note that the deformation grid is much coarser than the grid used for solving the Navier-Stokes equations. The user deforms the grid interactively and a deformed scalar stream function $\tilde{\psi}(t)$ is obtained. In this paper, we use a deformation method based on Moving Least Squares [Schaefer et al. 2006] to deform the stream function. Finally, we apply a curl operator ($\nabla\times$) to the deformed stream function $\tilde{\psi}(t)$, and the deformed velocity field $\tilde{\mathbf{v}}(t)$ is generated. Smoke densities are advected according to the deformed velocity field $\tilde{\mathbf{v}}(t)$ to visualize the flow. In the following subsections, we describe the details of each process.

4.1 Conversion to Stream Function

As described before, our method converts the velocity field $\mathbf{v}$ into the scalar stream function ψ. In order to explain this process, let us introduce the Helmholtz-Hodge Decomposition. In Helmholtz-Hodge Decomposition, any vector fields $\mathbf{w}$ can be decomposed into curl-free and divergence-free components as follows.

$$\mathbf{w} = \nabla\phi + \nabla\times\psi, \tag{4}$$

where ϕ is a scalar potential function and ψ is a scalar stream function. The first term on the right is the curl free component (i.e. $\nabla\times(\nabla\phi) = 0$) and the second term is the divergence-free component (i.e. $\nabla\cdot(\nabla\times\psi) = 0$). In this paper, in order to obtain a deformed divergence-free flow field, we represent the velocity field by using the scalar stream function, that is,

$$\mathbf{v} = \nabla\times\psi. \tag{5}$$

In general, a vector field basically consists of both a curl-free and a divergence-free components. Therefore, we cannot use Eq. 5 for arbitrary velocity fields. Fortunately, for the velocity field obtained by solving the Navier-Stokes equations consists of the divergence-free component only. This is because the pressure term in Eq.(1) is calculated using Helmholtz-Hodge Decomposition. We explain the reason for this in the following.

In solving the Navier-Stokes equations, we first compute an intermediate velocity field $\mathbf{w}_{tmp}$ by using Eq. 1 except for the pressure term. Next, to compute the pressure term (Eq.(1)), the velocity field $\mathbf{w}_{tmp}$ is decomposed using Helmholtz-Hodge Decomposition as shown in the following equation.

$$\mathbf{w}_{tmp} = \nabla p + \mathbf{v}, \tag{6}$$

First term on the right-hand side (∇p) is the curl-free component of the temporal velocity field and the second term represents divergence-free components. The pressure term is then computed computed by using the continuity equation (Eq. 2), and the velocity field is obtained from $\mathbf{w}_{tmp}$ and p (see [Stam 1999] for more details). That is,

$$\mathbf{v} = \mathbf{w}_{tmp} - \nabla p, \tag{7}$$

As indicated by the above equations, the velocity field obtained by solving the Navier-Stokes equations consists of the divergence-free component only. Thus, we can safely represent the velocity field $\mathbf{v}$ by using the stream function, $\nabla \times \psi$.

Next, we describe a method for calculating the scalar stream function ψ from the velocity field. ψ is calculated by using the Green function G (see [Li et al. 2006] for details) :

$$\psi(\mathbf{x}, t) = \sum_{\mathbf{y} \in \Omega} (\nabla \times G(\mathbf{x} - \mathbf{y})) \cdot \mathbf{v}(\mathbf{y}, t), \tag{8}$$

where $\mathbf{x}$ and $\mathbf{y}$ are grid points on the grid used for representing the velocity field (not the grid for deformation), Ω is the entire domain of the simulation space, $\cdot$ represents an inner product between two vectors. In this paper, we define the Green function G by the following equation :

$$G(\mathbf{x} - \mathbf{y}) = \exp(\frac{-||\mathbf{x} - \mathbf{y}||}{\sigma}), \tag{9}$$

where σ is a user-specified coefficient. The above equations indicate that the stream function is obtained by convolving the input velocity field $\mathbf{v}$ with a vector field represented as the curl of the Gaussian function.

4.2 Deformation of Flow Fields

In this subsection, we explain the deformation method. As described before, we prepare a deformation grid for deforming the stream function. We use the method based on the Moving Least Squares [Schaefer et al. 2006] for deforming the grid. The user places a set of control points $\mathbf{p}_i$ (the red points in Fig. 3) and the user moves these control points interactively to desired target positions $\mathbf{q}_i$. Then, the grid is deformed by using a deformation function $f_{\mathbf{r}}$, which is obtained by minimizing the following energy function E.

$$E = \sum_{i=0}^{N_p - 1} w_i ||f_{\mathbf{r}}(\mathbf{p}_i) - \mathbf{q}_i||, \tag{10}$$

where $\mathbf{r}$ is a grid point on the deformation grid (Fig. 3), $\mathbf{p}_i$ is the initial position of the i-th control point, and $\mathbf{q}_i$ is its target position.

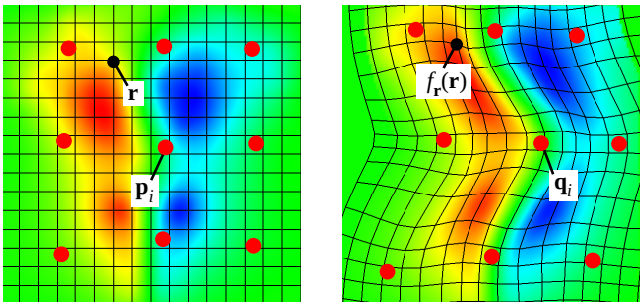


Figure 3: *Deformation grid.*

N_p is the number of control points placed by the user. The weight w_i is defined by the following equation.

$$w_i = \frac{1}{||\mathbf{p}_i - \mathbf{r}||^{\alpha}},$$

where α is a user-specified coefficient. For the details on solving the minimization problem, please see [Schaefer et al. 2006].

The scalar stream function is deformed by using the deformation function obtained by the above method. The deformation function is defined by using the deformation grid that is coarser than the simulation grid used for storing the stream function. The deformation function at each grid point of the simulation grid is therefore computed by the bilinear interpolation using the neighboring deformation grid points. This deformation is easily implemented by using the hardware texture mapping functions.

After the deformed stream function $\tilde{\psi}$ is calculated, the deformed velocity field $\tilde{\mathbf{v}}$ is obtained by applying curl operator to $\tilde{\psi}$.

$$\tilde{\mathbf{v}}(t) = \nabla \times \tilde{\psi}(t). \tag{11}$$

The resultant velocity field of course satisfies divergence-free condition since $\nabla \cdot \nabla \times \tilde{\psi}(t)$ is always zero by definition.

5 Results

This section shows some examples created by using our method. We used a desktop PC with an Intel Core i7 2600K CPU, 16GB memory to compute all the examples shown in this section. For all examples, the number of grid points for the simulation grid and the deformation grid was 128×128 and 64×64, respectively. The videos corresponding to the following examples can be found in the supplementary material.

Fig. 1 shows an example of rising smoke from a cartoon lamp. The rising smoke animation is generated by using the fluid simulation. The source of the smoke is located at the tip of lamp. Fig. 1(a) shows the original input smoke animation. Fig. 1(b) shows a result obtained by deforming the original flow field by using our method. In Fig. 1(a), smoke rises vertically. In Fig. 1(b), We deform the flow field to create a swirling smoke motion. By using our method, we can easily generate such interesting smoke motions

Fig. 4 shows a comparison of results with and without using the stream function for deforming the velocity field. Fig. 4(a) shows the original input flow field, and (b) shows the divergence of the velocity field at red rectangular domain in (a). Figs. 4(c) and (d) show results created by deforming the flow field by our method using the stream function. Figs. 4(e) and (f) are created by directly deforming the velocity fields by using Moving Least Squares, without using

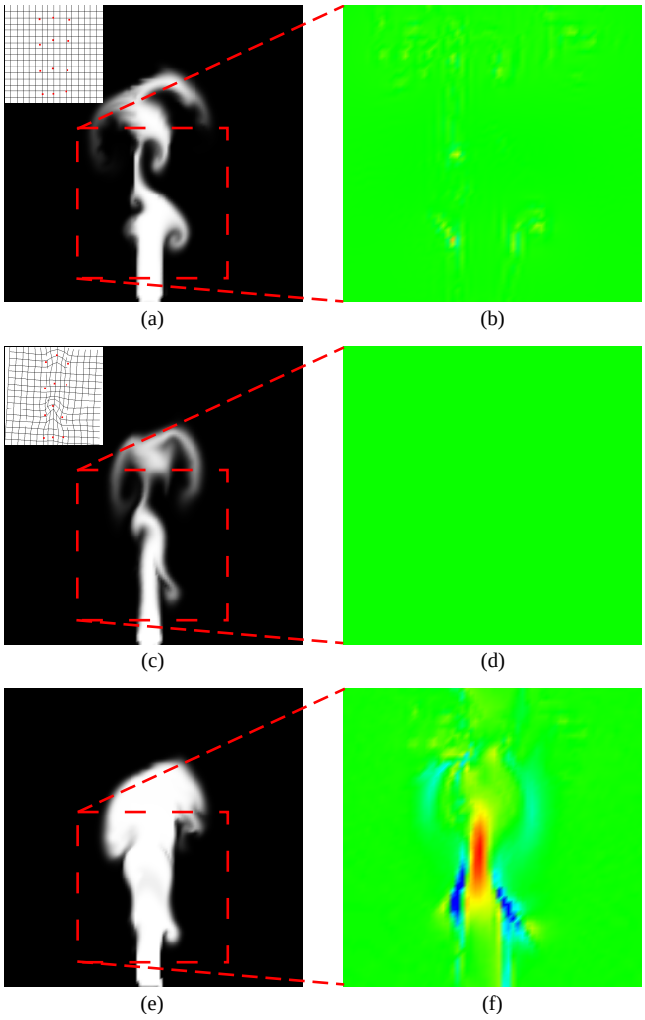


Figure 4: *Comparison of results.*

the stream function. In Figs. 4(b)(d)(f), the blue and red color indicate negative and positive values of divergence, respectively. The images shown at the left-upper corners in Figs. 4(a)(c)(e) show the deformation grid. In Figs. 4(c) and (e) , we deform the original flow field so that the width of the smoke becomes narrower. However, in Fig. (e), the width of smoke does not become narrow. This is due to the influence of high positive divergence shown in Fig. (f). In contrast, since the flow field generated by our method satisfies divergence-free condition, our method successfully generates the flow field satisfying the user's intention.

6 Conclusion

In this paper, we have proposed a method for deforming fluid flow while preserving the divergence-free condition. The divergence-free condition is enforced by converting the input flow field into the scalar stream function. The deformation is done by using the method based on the Moving Least Squares. We demonstrated the capabilities of our method with a set of examples.

As for future work, we are planning to extend our method to 3D flow fields. However, a stream function can not be defined other than axisymmetry flows in 3D fluid flow fields. Therefore, we cannot easily extend our method to 3D flow fields. We are planning to develop a method that approximates a 3D flow field by a set of scalar functions.

References

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

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.

LI, H., CHEN, W., AND SHEN, I.-F. 2006. Segmentation of discrete vector fields. *IEEE Transactions on Visualization and Computer Graphics 12*, 3, 289–300.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Transactions on Graphics 25*, 3, 533–540.

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.