Generating Flow Fields Variations by Modulating Amplitude and Resizing Simulation Space

Syuhei Sato* Hokkaido University
Yoshinori Dobashi Hokkaido University JST CREST
Kei Iwasaki Wakayama University
Hiroyuki Ochiai Kyushu University JST CREST
Tsuyoshi Yamamoto Hokkaido University

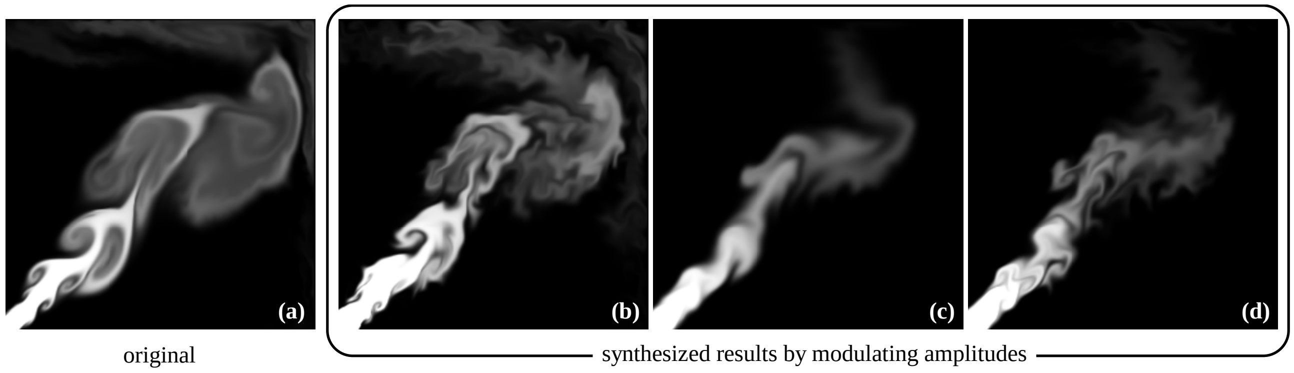

Figure 1: *Modulating amplitudes of a smoke animation by our method. (a) shows the input smoke animation. (b) through (d) show the results generated by modulating amplitudes of original animations.*

Abstract

The visual simulation of fluids has become an important element in many applications, such as movies and computer games. In these applications, large-scale fluid scenes, such as fire in a village, are often simulated by repeatedly rendering multiple small-scale fluid flows. In these cases, animators are requested to generate many variations of a small-scale fluid flow. This paper presents a method to help animators meet such requirements. Our method enables the user to generate flow field variations from a single simulated dataset obtained by fluid simulation. The variations are generated in both the frequency and spatial domains. Fluid velocity fields are represented using Laplacian eigenfunctions which ensure that the flow field is always incompressible. In generating the variations in the frequency domain, we modulate the coefficients (amplitudes) of the basis functions. To generate variations in the spatial domain, our system expands or contracts the simulation space, then the flow is calculated by solving a minimization problem subject to the resized velocity field. Using our method, the user can easily create various animations from a single dataset calculated by fluid simulation.

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

Keywords: flow field, variation synthesis, Laplacian eigenfunctions, amplitude modulation, resizing simulation space

*e-mail:sato@ime.ist.hokudai.ac.jp

1 Introduction

The visual simulation of fluids has become one of the most important research topics in computer graphics. Many methods have been proposed for simulating smoke, water, fire, etc [Stam 1999; Fedkiw et al. 2001; Nguyen et al. 2002; Feldman et al. 2003; 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. However, one of the problems is the expensive computational cost. In those entertainment applications, similar fluid animations are often required. Such examples include multiple explosions caused by missile attack, many flowing rivers, multiple houses on fire, and smoke rising from multiple chimneys. Using the same fluid animation repeatedly degrades the realism of a synthetic scene. Therefore, animators have to create multiple fluid animations with different motions. This is achieved by repeating the fluid simulation many times with different parameter settings. However, adjusting the simulation parameters to create such similar animations is very difficult and huge computational costs are required. Our research goal is to address this problem.

Procedural methods can generate similar multiple 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. Some researchers have developed procedural methods for modeling flow using turbulent noise functions [Patel and Taylor 2005; Bridson et al. 2007]. With these methods, animators can easily generate turbulent motion and multiple animations with similar motion can be created at low cost. However, these procedural methods do not use fluid simulation to generate the fluid flow, making the animations less realistic than those created using physically-based simulation.

To address this problem, we propose a method for generating various different fluid animations from a single simulated dataset. Our method is based on grid-based fluid simulation and is suitable for

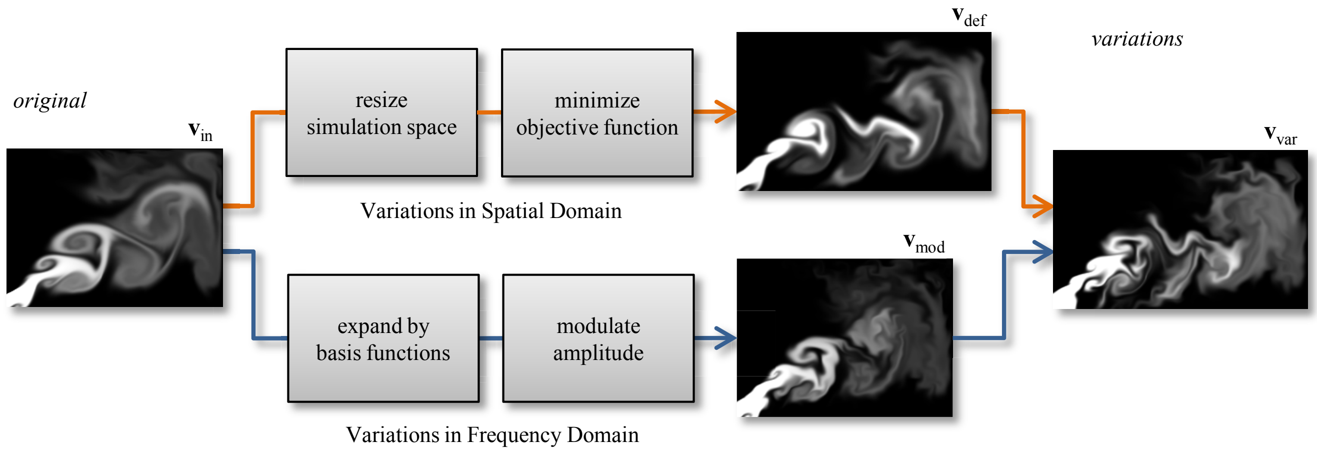
original
$\mathbf{v}_{\text{in}}$
resize simulation space
minimize objective function
Variations in Spatial Domain
expand by basis functions
modulate amplitude
Variations in Frequency Domain
$\mathbf{v}_{\text{def}}$
$\mathbf{v}_{\text{mod}}$
variations
$\mathbf{v}_{\text{var}}$

Figure 2: *Overview of our method.*

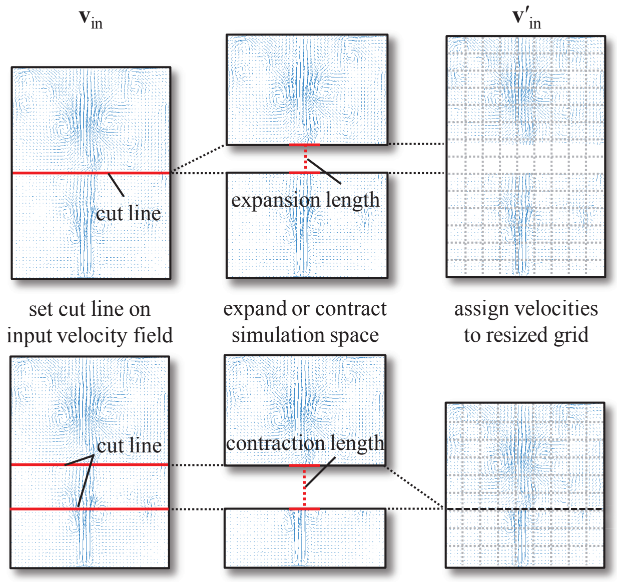
$\mathbf{v}_{\text{in}}$
$\mathbf{v}'_{\text{in}}$
cut line
expansion length
set cut line on input velocity field
expand or contract simulation space
assign velocities to resized grid
cut line
contraction length

Figure 3: *Expanding and contracting an input velocity field. Upper row shows how to expand the velocity field. Lower row shows contracting the velocity field.*

generating animations of divergence-free flow such as smoke and fire. Fig. 1 shows examples of the variations created by our method. The input data produces a simulated flow field of rising smoke from the bottom-left corner of the simulation space (Fig. 1(a)). Then, variations in the flow field are generated in both the frequency and spatial domains, where the user can specify the degree of variation. Using the resulting flow fields, scalar quantities such as the smoke density, are advected and the final images are created as shown in Figs. 1(b) through (d).

The key concept behind our method is to represent the input velocity fields using divergence-free basis functions in order to generate flow field variations. We use Laplacian eigenfunctions for the divergence-free basis functions [Witt et al. 2012]. The variations are generated by randomly modulating the coefficient of each frequency component. Furthermore, our system resizes the simulation space to generate variations in the spatial domain. The flow field in the resized space is calculated by solving a minimization problem. By combining these two methods, various fluid flows can be generated from a single simulated flow field.

2 The Method

Fig. 2 shows an overview of our method. Two types of processes are used to generate the variations. The first one is amplitude modulation which generates variations of the flow field $\mathbf{v}_{\text{mod}}(t_n)$ in the frequency domain. This process modulates the coefficients calculated by expanding the input flow field $\mathbf{v}_{\text{in}}(t_n)$ into divergence-free basis functions. The other one is resizing the simulation space which generates variations of the flow field $\mathbf{v}_{\text{def}}(t_n)$ in the spatial domain. In this process, the size of the simulation space is changed randomly. Our system then solves a minimization problem subject to the resized velocity fields, and generates the resultant flow fields. These processes can be combined and a flow field $\mathbf{v}_{\text{var}}(t_n)$ is generated. In the following subsections, we describe the details of the above two processes.

2.1 Variations in the Frequency Domain

First, we represent the input flow field $\mathbf{v}_{\text{in}}(t_n)$ by a linear combination of divergence-free basis functions $\mathbf{\Phi}_i$, that is,

$$\mathbf{v}_{\text{in}}(t_n) = \sum_{i=0}^{N-1} w_i(t_n)\mathbf{\Phi}_i, \tag{1}$$

where N is the number of basis functions, and t_n are discrete time steps $(n = 0, 1, \cdots)$. We use Laplacian eigenfunctions for the basis functions, $\mathbf{\Phi}_i$ [Witt et al. 2012]. This enforces the divergence-free condition on the flow field. The coefficient $w_i(t_n)$ for the i-th basis function is calculated using the following equation.

$$w_i(t_n) = \int_{\mathbf{x}\in\Omega} \mathbf{v}_{\text{in}}(t_n) \cdot \mathbf{\Phi}_i d\mathbf{x}, \tag{2}$$

where Ω is the entire domain of the input flow field and $\cdot$ is the dot product between two vectors.

By modulating $w_i(t_n)$, we generate variations of the flow fields $\mathbf{v}_{\text{mod}}(t_n)$ in the frequency domain. Our method modulates $w_i(t_n)$ for each frequency component as follows,

$$\mathbf{v}_{\text{mod}}(t_n) = \sum_{i=0}^{N-1} g_i w_i(t_n)\mathbf{\Phi}_i, \tag{3}$$

where g_i represents the gain with which $w_i(t_n)$ is modulated. In our method, g_i is generated randomly.

2.2 Variations in the Spatial Domain

In this subsection, we describe a method for generating flow field variations in the spatial domain. Our method changes the size of the simulation space by arbitrarily expanding or contracting it. Fig. 3 shows the method used to resize the simulation space. First, a direction and a line that cuts through the input velocity field are determined by random numbers. In Fig. 3, the cut direction is shown as being horizontal. Next, a distance to expand or contract the simulation space is determined, again using a random number. In the expansion process, the simulation space is divided by the cut line into two domains. These domains are separated and new grid points are inserted between them (Fig. 3). In the case of contraction, two cut lines are specified, and then the grid points between the two cut lines are removed as shown in Fig. 3. After resizing the simulation space, we prepare a new grid for the resized simulation space. The number of grid points for the new grid is N_{def}. The input velocities are copied to the new grid according to the resizing information of the simulation space. The resized velocity field is denoted by $\mathbf{v}'_{\text{in}}$. Then the coefficients $\mathbf{w}'(t_n) = (w'_0(t_n), w'_1(t_n), \cdots, w'_{N-1}(t_n))$ are calculated by solving the following minimization problem.

$$\arg\min_{\mathbf{w}'(t_n)} (E(t_n) + \alpha \sum_{i=0}^{N-1} w'^2_i(t_n)), \tag{4}$$

$$E(t_n) = \sum_{\mathbf{x}\in\Omega_{in}} |\mathbf{v}'_{\text{in}}(\mathbf{x}, t_n) - \sum_{i=0}^{N-1} w'_i(t_n)\mathbf{\Phi}'_i(\mathbf{x})|^2,$$

where Ω_{in} is the domain where the input velocities $\mathbf{v}_{\text{in}}$ are assigned on $\mathbf{v}'_{\text{in}}$, and α is a user-specified constant used to adjust the influence of the second term which is called the regularization term. $\mathbf{\Phi}'_i$ is defined on the grid of the resized simulation space. $E(t_n)$ measures the difference between the resized velocities and the resulting flow field. By solving the above equation, variations of the flow fields $\mathbf{v}_{\text{def}}(t_n)$ in the spatial domain are synthesized.

$$\mathbf{v}_{\text{def}}(t_n) = \sum_{i=0}^{N-1} w'_i(t_n)\mathbf{\Phi}'_i. \tag{5}$$

Given the definition of the error function $E(t_n)$ in the previous paragraph, we can now solve the minimization problem given in Eq. (4). By taking the derivative of Eq. (4) with respect to the coefficient $w'_i(t_n)$, we obtain the following matrix equation.

$$(\mathbf{A} + \alpha\mathbf{I})\mathbf{w}'(t_n) = \mathbf{c}(t_n), \tag{6}$$

where $\mathbf{A}$ and $\mathbf{I}$ are $N \times N$ matrices, and $\mathbf{w}'(t_n)$ and $\mathbf{c}(t_n)$ are N dimensional column vectors. $\mathbf{I}$ is the identity matrix. $\mathbf{A}$ and $\mathbf{c}$ are related to E. The (i, j)-th element a_{ij} of $\mathbf{A}$ and the i-th element c_i of $\mathbf{c}(t_n)$ are given respectively by:

$$\begin{aligned} a_{ij} &= \sum_{\mathbf{x}\in\Omega_{\text{in}}} \mathbf{\Phi}'_i(\mathbf{x}) \cdot \mathbf{\Phi}'_j(\mathbf{x}), \\ c_i &= \sum_{\mathbf{x}\in\Omega_{\text{in}}} \mathbf{v}'_{\text{in}}(\mathbf{x}, t_n) \cdot \mathbf{\Phi}'_i(\mathbf{x}), \end{aligned}$$

$\mathbf{A}$ is a full matrix as shown by the above equation. In order to compute the dot product between the functions in the above equations, $\mathbf{v}'_{\text{in}}$ and $\mathbf{\Phi}'_i$ are sampled on the grid of the resized simulation space. Then, an approximation of the dot product is computed using the sampled values.

Table 1: *Parameter settings and computation times. N is the number of basis functions, N_{def} is the number of grid points for resized velocity fields. T is the time for updating the flow field, measured in milliseconds.*

	N	N_{def}	T
Fig. 1	1024	–	120
Fig. 4(b)	400	64×96	10
Fig. 5(b)(c)	1024	256×320	140
Fig. 5(e)(f)	1024	256×192	80

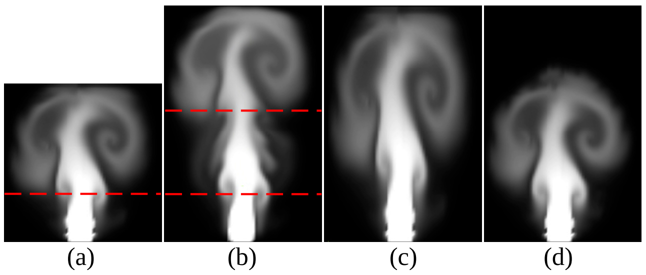

Figure 4: *Comparison of results using different methods. (a) original smoke simulated on a 64×64 grid. (b) flow computed our method. (c) flow warped from (a). (d) flow re-simulated with larger grid*

The coefficient vector $\mathbf{w}'(t_n)$ is then obtained by solving the matrix equation, Eq. (6). To solve the equation efficiently, we use the LU decomposition technique [Press et al. 2007]. The LU decomposition of the matrix $(\mathbf{A} + \alpha\mathbf{I})$ is firstly computed and the weight vector $\mathbf{w}'(t_n)$ is efficiently obtained using the decomposed matrix.

3 Results

This section shows some examples created using our method. We used a desktop PC with an Intel Core i7 2600K CPU, 16GB memory, and an NVIDIA GeForce GTX 680 GPU to compute all the examples shown in this section. The parameter settings and timing information are summarized in Table 1. For examples of Fig. 1 and Fig. 5, the number of grid points for the input velocity fields was 256×256. In Fig. 4, the input velocity field was computed on a 64×64 grid. The videos corresponding to the following examples can be found in the supplementary material.

Fig. 1 shows examples of the amplitude modulation. The flow field is visualized by advecting the smoke density. The source of the smoke is located at the bottom-left corner of the simulation space. Fig. 1(a) shows the input smoke animation. Figs. 1(b) through (d) show the variations created by modulating the amplitudes of the input animation. (b) is created by increasing the high-frequency components of the input velocity fields. In (c), the low-frequency components are reduced. (d) is created by applying the modulations of both (b) and (c). As shown in these examples, our method can generate variations in the frequency domain.

Fig. 4 shows a comparison of results generated by using the resizing approach. Fig. 4(a) shows the original flow field computed on a 64×64 grid. A cut line is shown by a red dotted line in Fig. 4(a). The result by using our method is shown in Fig. 4(b). Our method successfully created a continuous flow field and the flow is similar to the original one. Fig. 4(c) shows a flow field created by warping the original flow field into the resized simulation space. In this case, the flow is deformed from the original flow field. Fig. 4(d) shows a result obtained by re-simulating the flow field in the resized

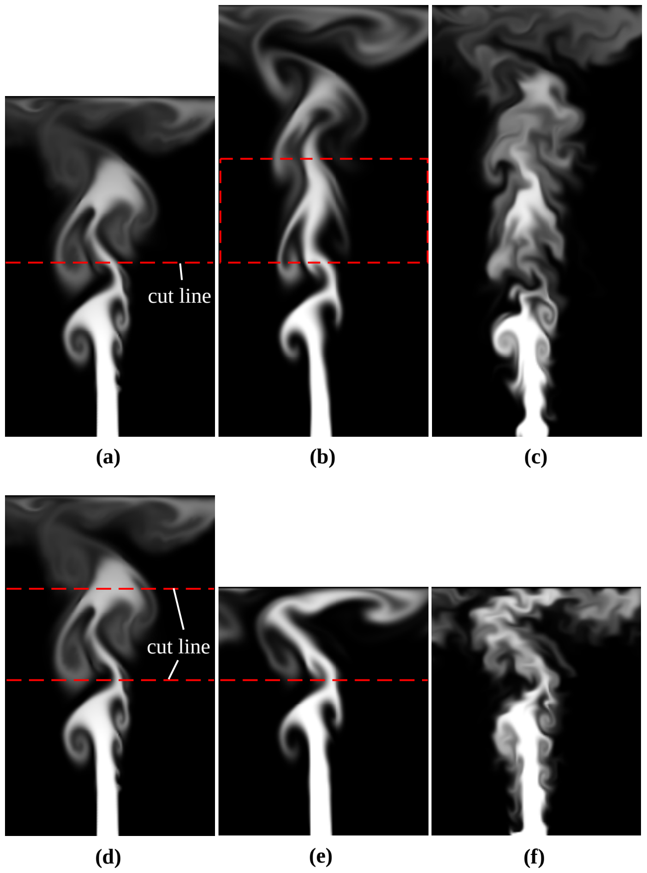

Figure 5: *Resizing simulation space of input smoke animation. (a) and (d) show an input smoke animations. In (b) and (e), a simulation space is resized by our method. (c) and (f) are variations in both the frequency and spatial domains.*

simulation space with the same parameter settings as in Fig. 4(a). In this case, the smoke does not rise into the higher region unless we adjust the simulation parameters adequately through a trial-and-error process.

Fig. 5 shows the results obtained using our method. Fig. 5(a) and (d) show the original flow field. The cut lines in Figs. 5(a) and (d) are shown by red dotted lines. In Figs. 5(b) and (c), the simulation space has been expanded. In Figs. 5(e) and (f), the simulation space has been contracted. The amount of expansion and contraction is set to a quarter of the vertical height of the input simulation space. Furthermore, Figs. 5(c) and (f) are created by additionally applying amplitude modulation so that the high-frequency components are enhanced. Using our method continuous flow fields can be successfully created. In addition, we can generate variations in both the frequency and spatial domains.

4 Conclusion

We have proposed a method for synthesizing variations of flow fields. Our method can generate variations in both the frequency and spatial domains. The flow fields are represented by Laplacian eigenfunctions, and the variations in the frequency domain are generated by modulating the coefficients of the basis functions. In addition, we can generate variations in the spatial domain by solving a minimization problem after resizing the velocity field. We demonstrated the capabilities of our method with a set of examples. As our future work, we are planning to extend our method to 3D flow fields.

One of the limitations of our method is that the computational cost is proportional to the number of basis functions. The level of detail in the flow generated by our method depends on the number of basis functions used. If the number of basis functions is large, high computational costs are required to calculate $\mathbf{A}$, $\mathbf{c}(t_n)$ and to reconstruct the flow fields using Eqs. (3) and (5). The user can generate variations of detailed flow fields at the expense of increased computational and storage costs.

Another limitation is the fact that the flow fields generated by our method might not conform to the laws of fluid flow, if the degree of modulation by g_i is too large. To address this problem, we calculate the Navier-Stokes equations for velocity fields generated by our method, then compare two velocity fields. By this comparison, we can evaluate whether or not the flow generated by our method follows the laws of fluid flow. In future work, we will make experiments to evaluate these flow fields.

Acknowledgements

We would like to thank UEI Research for their support.

References

Bridson, R., Hourihan, J., and Nordenstam, M. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics 26*, 3, Article 46.

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.

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.

Lamorlette, A., and Foster, N. 2002. Structural modeling of flames for a production environment. *ACM Transactions on Graphics 21*, 3, 729–735.

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.

Patel, M., and Taylor, N. 2005. Simple divergence-free fields for artistic simulation. *Journal of Graphics, GPU, and Game Tools 10*, 4, 49–60.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 2007. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press.

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

Witt, T. D., Lessig, C., and Fiume, E. 2012. Fluid simulation using laplacian eigenfunctions. *ACM Transactions on Graphics 31*, 1, Article 10.