Stream-Guided Smoke Simulations

SYUHEI SATO, University of Toyama, Prometech CG Research, JAPAN
YOSHINORI DOBASHI, Hokkaido University, Prometech CG Research, JAPAN
THEODORE KIM, Yale University, USA

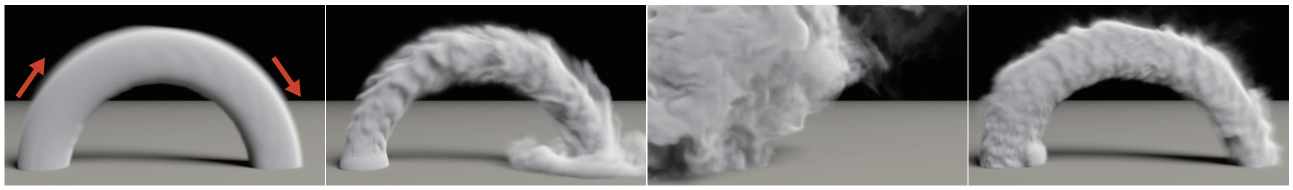
(a) input (b) ours $\alpha = 0.01$, $c_{vc} = 5.0$ (c) frequency-domain $\nu_{cutoff} = 1/2$, $c_{vc} = 5.0$ (d) variational guiding $\phi_{low} = 0.0$, $\phi_{high} = 0.35$, $c_{vc} = 5.0$

Fig. 1. The procedurally-generated input (a). Figures (b) (c) and (d) respectively show results with our method, the frequency-domain [Forootaninia and Narain 2020], and variational guiding method [Nielsen and Christensen 2010]. The α is a control parameter for our guiding method, c_{vc} is a vorticity confinement coefficient. ν_{cutoff} is the cutoff frequency of the frequency-domain method. ϕ_{low} and ϕ_{high} are the control parameters of the variational method.

High-resolution fluid simulations are computationally expensive, so many post-processing methods have been proposed to add turbulent details to low-resolution flows. Guiding methods are one promising approach for adding naturalistic, detailed motions as a post-process, but can be inefficient. Thus, we propose a novel, efficient method that formulates fluid guidance as a minimization problem in *stream function space*. Input flows are first converted into stream functions, and a high resolution flow is then computed via optimization. The resulting problem sizes are much smaller than previous approaches, resulting in faster computation times. Additionally, our method does not require an expensive pressure projection, but still preserves mass. The method is both easy to implement and easy to control, as the user can control the degree of guiding with a single, intuitive parameter. We demonstrate the effectiveness of our method across various examples.

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

Additional Key Words and Phrases: fluid simulation, upsample, guiding flow, stream function

ACM Reference Format:
Syuhei Sato, Yoshinori Dobashi, and Theodore Kim. 2021. Stream-Guided Smoke Simulations. *ACM Trans. Graph.* 40, 4, Article 161 (August 2021), 7 pages. https://doi.org/10.1145/3450626.3459846

1 INTRODUCTION

Fluid simulations are ubiquitous in film and game production, but remain computationally expensive and tedious to control, requiring users to time-consumingly and painstakingly tweak many parameters. Many methods have been proposed to accelerate this process, including control forces, direct editing, and physics-based post-processing. In the paper, we present a guide-based post-processing method, as these approaches generally generate the most naturalistic flows by finding solutions that holistically satisfy natural laws. However, these guide-based methods [Nielsen and Christensen 2010; Nielsen et al. 2009] also usually solve large minimization problems, which can make their computational cost quite large.

We present a more efficient method that works on *stream functions* (a.k.a. *vector potentials* in 3D). This significantly reduces the problem size, and allows the computations to complete much more quickly than previous methods [Nielsen and Christensen 2010; Nielsen et al. 2009], while still yielding high-quality, high-resolution flows. Our method converts arbitrary input velocities into stream functions, that are then used to guide the optimization of a higher-resolution stream. The final velocity can be retrieved by simply taking the curl, and the result is guaranteed to be incompressible. Similar to previous methods, we formulate a minimization problem, but our stream formulation reduces the complexity of the problem by phrasing it as the scalar product of the upsampled input stream function and a simple *scaling* function. The goal is then to obtain the optimal scaling field that generates a high-resolution turbulent flow that is similar to the input flow. The problem complexity is 16 times smaller than that of previous methods. Our method inherits the attractive property of optimization-based approaches where results are achieved faster than brute-force solves, while still maintaining the incompressibility in the final flow. Fig. 1 shows an example of our method.

Relative to previous noise-based [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008] or data-driven approaches [Chu and Thuerey 2017; Sato et al. 2018], our approach produces more naturalistic results. Compared to very recent frequency-based guiding work [Forootaninia and Narain 2020], our method is robust to non-physical inputs, and is able to generate convincing results even when the input velocities were generated procedurally, or hand-drawn by the user. This makes our method more widely applicable

Authors' addresses: Syuhei Sato, University of Toyama, Prometech CG Research, Toyama, JAPAN, ssato@eng.u-toyama.ac.jp; Yoshinori Dobashi, Hokkaido University, Prometech CG Research, Sapporo, JAPAN, doba@ime.ist.hokudai.ac.jp; Theodore Kim, Yale University, New Haven, USA, kim@cs.yale.edu.

to artistic flow designs, and not limited to existing low-resolution physics-based simulations.

2 RELATED WORK

Stam [1999] introduced the first unconditionally stable solver for the Navier-Stokes equations to computer graphics. Since then, many methods have been proposed for simulating fluid phenomena, which are summarized in many excellent texts [Bridson 2015; Kim 2017]. Realistic animations can be produced with these methods, but users must still run repeated simulations and search for good parameter settings in order to obtain a desired motion. We discuss previous approaches to this problem here.

Several control methods have been proposed for creating a desired fluid motion. Fattal and Lischinski [2004] introduced additional external forces to control smoke simulations, while Thürey et al. [2006] proposed a detail-preserving control method. This work decomposes a fluid velocity field into multi-scale components, and then only applies control forces to coarse-scale components. Although these control methods can create desired fluid motions, multiple costly high-resolution simulations are needed to design an overall motion.

Many methods have been proposed for enhancing flow detail as a post-process. Some of them synthesize plausible turbulent motion for a low-resolution simulation using wavelet noise [Kim et al. 2008] or curl noise [Narain et al. 2008; Schechter and Bridson 2008]. In recent years, existing high-resolution flow data has been used for detail enhancement. Chu and Thuerey [2017] used a Convolutional Neural Network to learn a relation between low- and high-resolution flows. Sato et al. [2018] introduced style transfer for turbulence, where details are transferred to low-resolution flows using patch-based texture synthesis. Although these approaches synthesize plausible details, the use of noise functions or other data results in flows that are less realistic than those from direct, high-resolution simulations.

Several guiding methods that work on velocity fields have also been proposed for enhancing detail. Nielsen et al. [2010; 2009] proposed methods for synthesizing guided high-resolution flows using an optimization-based approach. This method constrains the low-frequency components of a high-resolution simulation to follow an input guide field. The degree of guiding is controlled by a user-specified coefficient. Although this method is useful, the problem is formulated as a minimization problem that satisfies the incompressibility condition, which results in a large, asymmetric, computationally expensive matrix. Gregson et al. [2014] reported a guided simulation based on an ADMM framework, while Inglis et al. [2017] also proposed an optimization-based approach to guide simulation via a primal-dual algorithm. However, this method also requires relatively high computation.

Forootaninia and Narain [2020] proposed a frequency-domain guiding method to address these problems. Their method uses an input guiding velocity field and the corresponding high resolution velocity field. These velocity fields are transformed into the frequency domain and the low frequency components of the high resolution field are replaced with those of the guiding velocity field. The synthesized frequency components are then transformed back to the spatial domain to obtain the guided velocity field, and a pressure projection step is then performed. This method is simple to implement and efficient compared to the previous methods, but requires the user to prepare a high resolution velocity field that is similar to the input guiding velocity field, which can be a difficult task for procedurally-generated guiding velocity fields.

We generate guided high-resolution flows using stream functions, which have been used in the past for several different purposes. For example, Ando et al. [2015] proposed a liquid solver which is performed in stream function space, and Sato et al. [2015] introduced a stream function for deforming fluid flow as a post-process. Input flows were converted into stream functions and then deformed. These approaches have the advantage that the results always satisfy the incompressibility condition, and our method shares this property.

3 PROPOSED METHOD

In the following, we use lower case letters to denote quantities on a low-resolution grid (e.g. $\mathbf{u}_t$), and upper case letters for those on a high-resolution grid (e.g. $\mathbf{U}_t$).

An overview of our method is shown in Fig. 2. At each frame of the animation, our method takes as input a low resolution guiding velocity field $\mathbf{u}_t$, and upsamples it to a high resolution velocity field $\mathbf{U}_t$. The upsampled velocity field is converted to a stream function $\mathbf{\Psi}_t$. An optimization problem is solved to generate a guided stream function $\hat{\mathbf{\Psi}}_t$, and the final velocity field $\hat{\mathbf{U}}_t$ is obtained by taking the curl of $\hat{\mathbf{\Psi}}_t$. This velocity field is then advected by solving the momentum equation of the Navier-Stokes equations and used to solve the optimization problem at the next frame of animation. We can optionally add external forces, such as vorticity confinement forces that increase turbulence. Our method allows the user to control the degree of guiding with a single parameter α. The details of these processes follow.

3.1 Computing the Stream Function

Using the Helmholtz-Hodge decomposition [Bhatia et al. 2013; Tong et al. 2003], $\mathbf{U}_t$ can be decomposed into:

$$\mathbf{U}_t = \nabla \times \mathbf{\Psi}_t + \nabla P + \mathbf{H}, \tag{1}$$

where P is a scalar function, $\mathbf{H}$ is a harmonic vector field. By applying $\nabla\times$ to the both sides of the equation, we obtain:

$$\nabla \times (\nabla \times \mathbf{\Psi}_t) = \nabla \times \mathbf{U}_t. \tag{2}$$

Next, by using the following relationship,

$$\nabla \times (\nabla \times \mathbf{\Psi}_t) = \nabla(\nabla \cdot \mathbf{\Psi}_t) - \nabla^2 \mathbf{\Psi}_t, \tag{3}$$

and $\nabla \cdot \mathbf{\Psi}_t = 0$ as a constraint, we obtain:

$$-\nabla^2 \mathbf{\Psi}_t = \nabla \times \mathbf{U}_t. \tag{4}$$

We can solve the above equation to compute $\mathbf{\Psi}_t$ from $\mathbf{U}_t$. We used the conjugate gradient method (CG) as the numerical solver. Ando et al. [2015] and Sato et al. [2015] provide more details.

3.2 Guiding Formulation

We will first describe our guided stream function, followed by our optimization problem. We represent the guided stream function $\hat{\mathbf{\Psi}}_t$

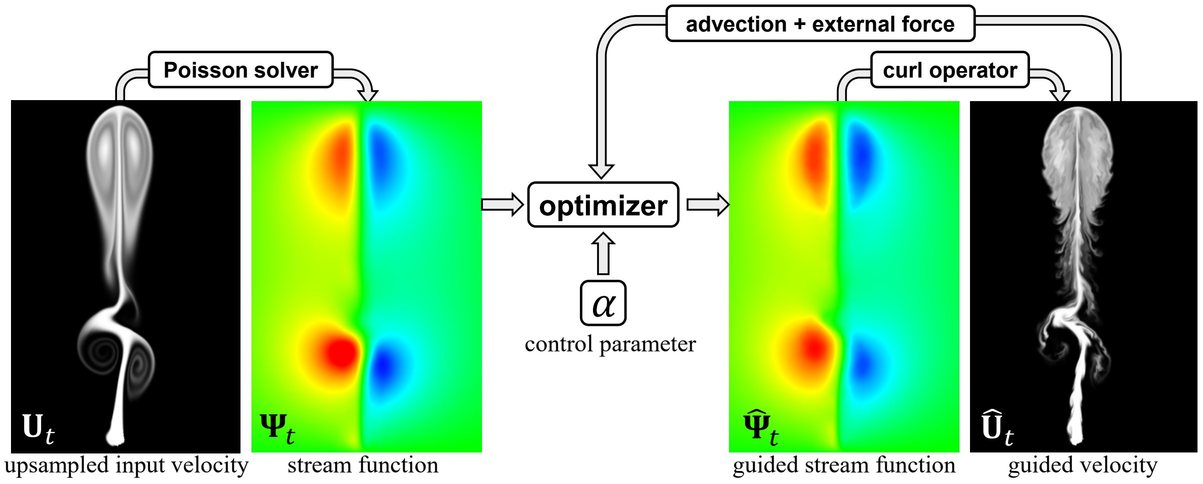

Fig. 2. Overview of our method.

as the sum of an upsampled stream function $\mathbf{\Psi}_t$ and any stream function from Section 3.1 $\tilde{\mathbf{\Psi}}_t$,

$$\hat{\mathbf{\Psi}}_t(\mathbf{x}) = \mathbf{\Psi}_t(\mathbf{x}) + \tilde{\mathbf{\Psi}}_t(\mathbf{x}), \tag{5}$$

where $\mathbf{x}$ is a grid position. We can then optimize $\tilde{\mathbf{\Psi}}_t(\mathbf{x})$ to generate high-frequency turbulent details. In lieu of directly computing $\tilde{\mathbf{\Psi}}_t(\mathbf{x})$, we instead represent it as the product of the scaling function $w_t(\mathbf{x})$ and an user-provided stream function $\mathbf{\Psi}_{\mathrm{usr},t}(\mathbf{x})$:

$$\tilde{\mathbf{\Psi}}_t(\mathbf{x}) = w_t(\mathbf{x})\mathbf{\Psi}_{\mathrm{usr},t}(\mathbf{x}). \tag{6}$$

We then optimize the scaling function, which reduces the complexity of the optimization problem by a factor of three: $w(\mathbf{x})$ is three times smaller than $\tilde{\mathbf{\Psi}}_t(\mathbf{x})$. This method additionally allows the user to choose any arbitrary $\mathbf{\Psi}_{\mathrm{usr},t}(\mathbf{x})$ obtained via the process of Section 3.1. However, based on our experiments, we found that using the input stream function, $\mathbf{\Psi}_t$, produced plausible high resolution flows.

We can now define the optimization problem. Our guiding formulation is similar to the previous optimization-based methods [Nielsen and Christensen 2010; Nielsen et al. 2009], but instead uses the stream functions described above.

We solve for the optimal scaling function subject to the following minimization problem at each frame of the animation,

$$\underset{w_t(\mathbf{x})}{\arg\min} \sum_{\mathbf{x}} ||\nabla \times \hat{\mathbf{\Psi}}_t(\mathbf{x}) - \mathbf{V}_t(\mathbf{x})||^2 + \alpha(w_t(\mathbf{x}))^2, \tag{7}$$

where α controls the degree of guiding. $\mathbf{V}_t(\mathbf{x})$ is obtained by advecting the guided velocity field from the previous frame. This is computed as

$$\frac{\partial \mathbf{V}_t}{\partial t} = -(\mathbf{V}_t \cdot \nabla)\mathbf{V}_t + \mathbf{f}, \tag{8}$$

and, by replacing $\mathbf{V}_{t-1}$ with $\hat{\mathbf{U}}_{t-1}$, we obtain the following discrete form:

$$\frac{\mathbf{V}_t - \hat{\mathbf{U}}_{t-1}}{\Delta t} = -(\hat{\mathbf{U}}_{t-1} \cdot \nabla)\hat{\mathbf{U}}_{t-1} + \mathbf{f}, \tag{9}$$

where Δt is a time step, and $\mathbf{f}$ is external forces such as vorticity confinement and smoke sources. Eq. (9) is a momentum equation in Navier-Stokes equations. In our implementation, the advection term (the first term on the right hand side of Eq. (9)) is solved using a semi-Lagrangian scheme [Stam 1999], but our method does not depend on a specific advection scheme, so it can be freely selected. The $\hat{\mathbf{U}}_{t-1}(\mathbf{x})$ term is:

$$\hat{\mathbf{U}}_{t-1}(\mathbf{x}) = \nabla \times \hat{\mathbf{\Psi}}_{t-1}(\mathbf{x}). \tag{10}$$

We can ignore the pressure and divergence-free terms from the Navier-Stokes equations because our stream functions already guarantee that the flow will be divergence-free.

We can interpret our optimization problem as follows. By using Eqs. (5) and (6), the first term of Eq. (7) can be rewritten as:

$$||\nabla \times (w_t(\mathbf{x})\mathbf{\Psi}_t) - (\mathbf{V}_t(\mathbf{x}) - \nabla \times \mathbf{\Psi}_t(\mathbf{x}))||^2. \tag{11}$$

The second term in the above equation is the difference between the upsampled input velocity field, $\nabla \times \mathbf{\Psi}_t$, and the simulated velocity field, $\mathbf{V}_t$, which was generated by the advection operator and external forces. The above equation tries to find the optimal scaling function w_t that reproduces the turbulent motion of $\mathbf{V}_t$. The second term of Eq.(7) works as a regularization term whose strength is controlled by α. Large α strongly constrains the output to follow the input flow, and suppresses turbulent motion, while the converse is true for small α.

The stream function has a non-trivial null space [Ando et al. 2015]. Our optimization therefore picks one of the possible solutions that minimizes $\alpha(w_t(\mathbf{x}))^2$, such that the solution is unique. However, Ψ_{usr} contains some ambiguity because w_t can vary depending on the choice of Ψ_{usr}, even if the final velocity is the same. Poor choices for Ψ_{usr} can produce unpleasant flows. In our experiments, we found that the stream functions from Section 3.1 should be used as Ψ_{usr} to obtain plausible flows. We discuss further in Section 5.

The minimization problem can be solved by taking the derivative of the objective function with respect to $w_t(\mathbf{x})$. This yields a linear system that we then solve using the conjugate gradient method. After solving the optimization problem, we obtain the guided potential $\hat{\mathbf{\Psi}}_t$ and the final velocity field $\hat{\mathbf{U}}_t$ is given by taking the curl of the

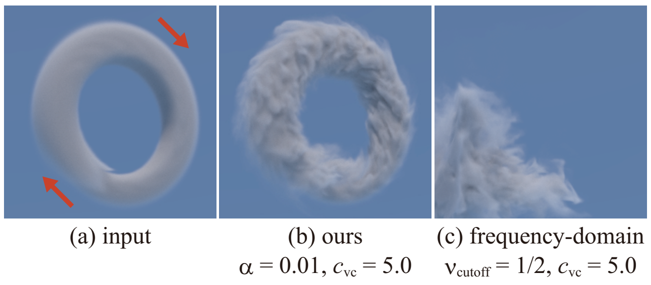
(a) input
(b) ours
$\alpha = 0.01$, $c_{vc} = 5.0$
(c) frequency-domain
$\nu_{cutoff} = 1/2$, $c_{vc} = 5.0$

Fig. 3. Result with a procedurally-generated flow in the shape of a ring.

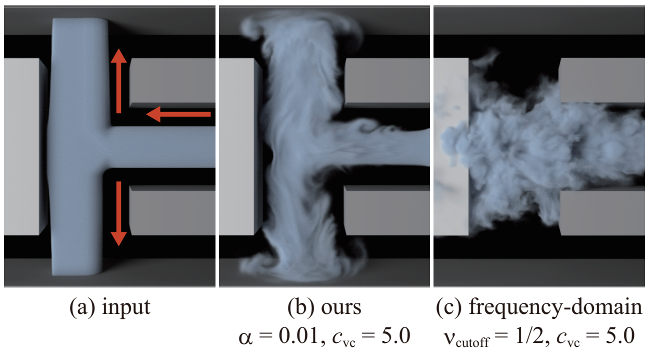
(a) input
(b) ours
$\alpha = 0.01$, $c_{vc} = 5.0$
(c) frequency-domain
$\nu_{cutoff} = 1/2$, $c_{vc} = 5.0$

Fig. 4. Result with a procedurally-generated branching flow.

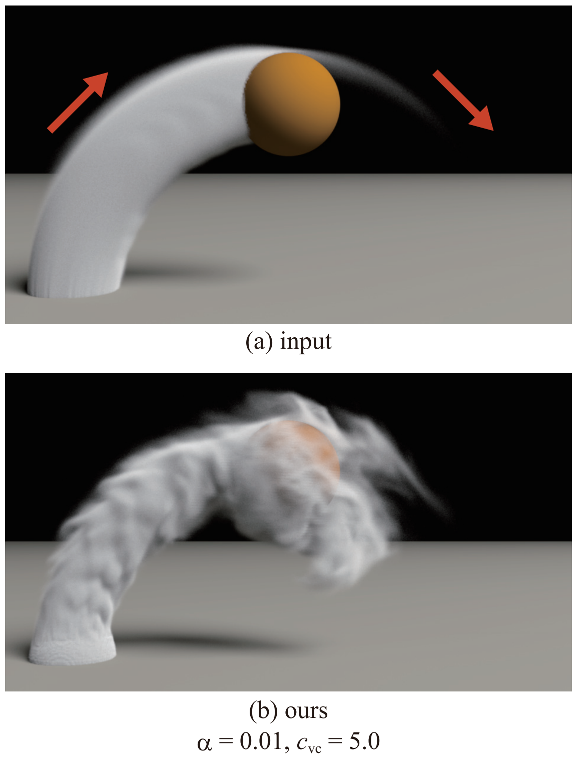

Fig. 5. Smoke arch interacting with a sphere.

guided potential. By definition, $\hat{\mathbf{U}}_t$ satisfies the incompressibility condition.

3.3 Using Control Parameter α

The regularization coefficient in Eq. (7), α, varies throughout space, and makes it possible to generate spatially-varying turbulence. The user can paint α according to the desired levels of turbulence.

Additionally, we found that our solver for Eq. (7) converged faster for larger α, which suggested that the computation can be accelerated by adaptively choosing an appropriate α at each grid point. Since the flow is visualized by a smoke density, and turbulent details are only visible in regions with non-zero densities, we assigned small α to the grid points in the smoke region and large values to other regions. While the ratio depends on the size of the smoke, this resulted in an average of 10% reduction in computation time. Since a sudden change in α at the boundary of the smoke region can cause artifacts, we linearly interpolated the value in this region.

4 RESULTS

In this section, we show examples and compare against the frequency-domain method of Forootaninia and Narain [2020] and the variational guiding method of Nielsen and Christensen [2010]. Figs. 6 and 8 use input velocity fields that were calculated using a semi-Lagrangian solver [Stam 1999], while other figures used inputs generated through alternate means. We used a semi-Lagrangian advection scheme, and vorticity confinement was used in calculating Eq.(9) to add more turbulence, where its amount is adjusted with a coefficient c_{vc} which is equivalent to ϵ from Fedkiw et al. [2001]. We used a desktop PC with an Intel Core i9-9900K CPU to compute all examples. The grid sizes, parameters, and computation times are summarized in Table 1. For the artistic examples (Figs. 1, 3 - 5, and, 7), we used static input velocity fields. For such input flows, our method only requires the conversion from stream function to velocity field to be computed once. For the frequency-based method, the accuracy of the pressure projection step was set so that its computation time would closely match the running time of our algorithm. The videos of these examples can be found in the supplementary material.

We first show several examples of procedurally-generated input flows that demonstrate the advantages of our optimization-approach. For all the inputs in these examples, velocities and densities are only assigned in regions of interest. In Fig. 1, the input flow forms an arch, and does not conform to any physical laws. Regardless, our method successfully synthesizes realistic turbulent motion (Fig. 1(b)) while the frequency-domain method struggles (Fig. 1(c)). The frequency-domain approach successfully addresses one limitation of optimization-based approaches, which is that they tend to blur high-frequency details. However, for the procedural input used in this experiment, velocities are only assigned in specific regions, which creates severe discontinuities at the region boundaries. These discontinuities cannot be accurately represented using low-frequency velocity fields, which renders the frequency-domain method unworkable. Fig. 1(d) is created using the variational guiding method. This can synthesize turbulent motion, but it is less

Table 1. Simulation statistics.

Figure	grid size		pre-computation time [sec]	computation time/frame [sec]		
	input	result		optimization	previous methods	unguided simulation
Fig. 1	$64 \times 32 \times 32$	$256 \times 128 \times 128$	124 (once)	28	(c)24, (d)203	28
Fig. 3	$48 \times 48 \times 32$	$192 \times 192 \times 128$	146 (once)	41	36	34
Fig. 4	$48 \times 48 \times 32$	$192 \times 192 \times 128$	146 (once)	29	28	34
Fig. 5	$64 \times 32 \times 32$	$256 \times 128 \times 128$	124 (once)	38	–	28
Fig. 6	64×64	512×512	3.4 (per frame)	0.92	–	2.1
Fig. 7	$24 \times 48 \times 24$	$192 \times 384 \times 192$	599 (once)	176	–	152
Fig. 8	$24 \times 64 \times 24$	$192 \times 512 \times 192$	702 (per frame)	(b)123, (c)151 (f)118, (g)150	(d)163, (e)140 (h)134, (i)126	220

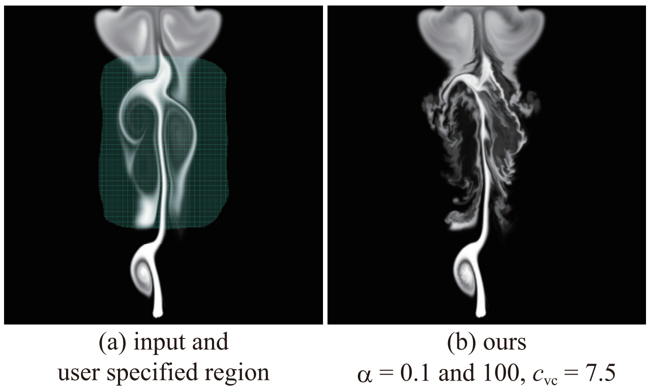

Fig. 6. 2D smoke with spatially-varying α.

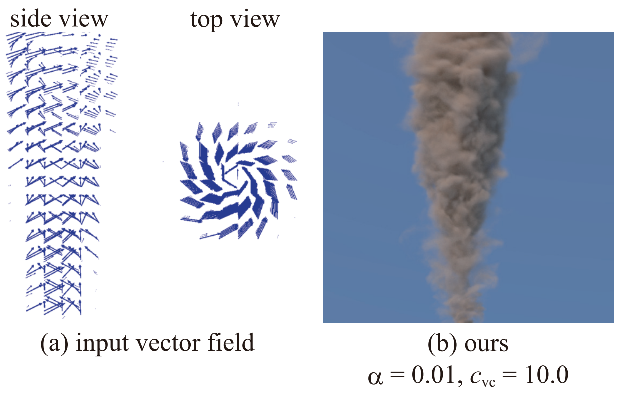

Fig. 7. Result from whirlwind-like procedural flow.

realistic than ours. Furthermore, the computation time for the optimizing velocity and pressure fields is seven times longer than ours (see Table 1). Note that our implementation of the variational method has neither parallel computation nor a multigrid solver. Thus, our timings are longer than those reported in their paper. Due to their larger problem dimensions, various optimizations are needed to achieve the best performance. In contrast, our optimization-based method synthesizes guided flows that satisfy physical laws across all frequency bands, and result in more naturalistic flows. Furthermore, our method does not need the various optimization techniques present in the variational method. This feature is particularly important for artistic flow designs in production environments.

Figs. 3 and 4 respectively show procedural examples of circular and branching smoke. Both input flows were created manually, and the outputs are essentially impossible to create using physically-based simulation alone, but our method is able to synthesize realistic-looking results. Fig. 5 uses the same input flow as Fig. 1, but places a sphere obstacle at the midpoint. In the input, the smoke does not interact with the sphere in any physically-plausible way; we simply assigned the velocity to zero inside the sphere. After converting the input velocity field into a stream function, we applied the method proposed in Bridson et al. [2007] to consider the effect of the sphere. Our method then successfully synthesizes a realistic flow that plausibly interacts with the sphere in Fig. 5 (b).

Next, we show an example demonstrating the effect of the parameter α (see Fig. 6). We can spatially vary the degree of turbulence by spatially modulating α. In Fig. 6 (a), α is 0.1 inside the green region but it is 100 elsewhere. Fig. 6 (b) shows the result: turbulent details only appear inside the green region.

Fig. 7 shows a more artistic example where the input flows were created manually. In Fig. 7, we created a whirlwind-like flow, and our system then adds realistic motion to this artistic flow.

Finally, we use an input flow that was computed by solving the Navier-Stokes equation (Fig. 8). We show smoke computed by our method with various parameter settings. Fig. 8 (a) shows the input flow simulating a smoke plume. Fig. 8 (b), (c), (f), and (g) show the guided high resolution flows, and the parameter values are shown in the captions. The semi-Lagrangian scheme was used to compute the advection term. These examples use the same stream function converted from the input velocity field, but since we do not have to re-compute the stream function every time the parameters are changed, so parameter tuning can be done efficiently. These examples demonstrate the effects of the parameters: detailed turbulent motions are likely to be synthesized with smaller α and larger c_{VC}. Fig. 8(d), (e), (h), and (i) show the same scene computed by the frequency-domain method [Forootaninia and Narain 2020]. The cutoff frequency is set to the value shown in the caption. Although high-frequency details are generated, the results look noisy (see the supplemental video).

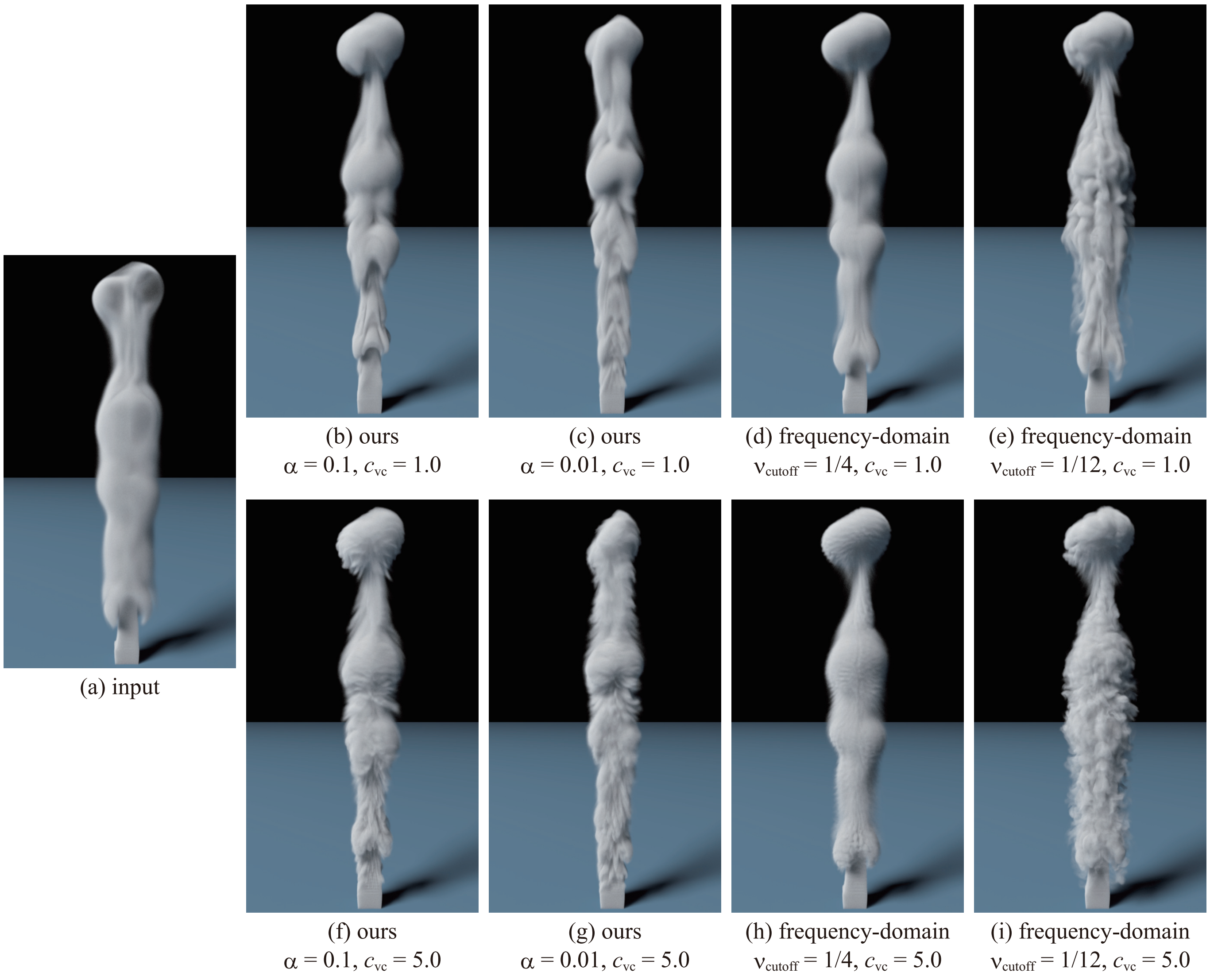
(a) input
(b) ours
$\alpha = 0.1$, $c_{vc} = 1.0$
(c) ours
$\alpha = 0.01$, $c_{vc} = 1.0$
(d) frequency-domain
$\nu_{cutoff} = 1/4$, $c_{vc} = 1.0$
(e) frequency-domain
$\nu_{cutoff} = 1/12$, $c_{vc} = 1.0$
(f) ours
$\alpha = 0.1$, $c_{vc} = 5.0$
(g) ours
$\alpha = 0.01$, $c_{vc} = 5.0$
(h) frequency-domain
$\nu_{cutoff} = 1/4$, $c_{vc} = 5.0$
(i) frequency-domain
$\nu_{cutoff} = 1/12$, $c_{vc} = 5.0$

Fig. 8. Plume examples with different parameters.

5 DISCUSSION

Our guiding fails when an inappropriate $\mathbf{\Psi}_{\text{usr}}$ is chosen. We investigated what constitutes an acceptable stream functions for $\mathbf{\Psi}_{\text{usr}}$ using 2D experiments over various $\mathbf{\Psi}_{\text{usr}}$. First, a $\mathbf{\Psi}_{\text{usr}} = 0$ function (obviously) did not work, and yielded the input flow. The functions $\mathbf{\Psi}_{\text{usr}} = 1$ and $\mathbf{\Psi}_{\text{usr}} = \mathbf{\Psi}_t + C$, where C is any constant value, also did not work, as the resulting flows were completely different from the input, and contained spurious details along with the domain boundaries (see Fig. 9 (b), (c)). We experimented with other $\mathbf{\Psi}_{\text{usr}}$, including noise functions and images of smoke, but the resultant flows around the boundaries were unacceptable (see Fig. 9 (d), (e) and the supplemental video). In contrast, the $\mathbf{\Psi}_{\text{usr}}$ obtained from Section 3.1 produced the convincing results shown in Section 4. These experiments suggest that plausible flows arise from the stream functions computed by Section 3.1.

Our guided flows always satisfy incompressibility by solving the optimization in stream function space. However, the divergent and harmonic components vanish from the input velocity fields during the conversion process from Section 3.1, since the curl of both terms in Eq.(1) is always zero by construction. However, if we factor the harmonic component from the input, it can be later added back. We experimented with a harmonic vector field as the input, and the stream function $\mathbf{\Psi}_{\text{usr}}$ obtained from another non-harmonic velocity field via Section 3.1. After the optimization, the harmonic component is simply added to the velocity. By doing this, a harmonic vector field can be used as a guiding input of our system.

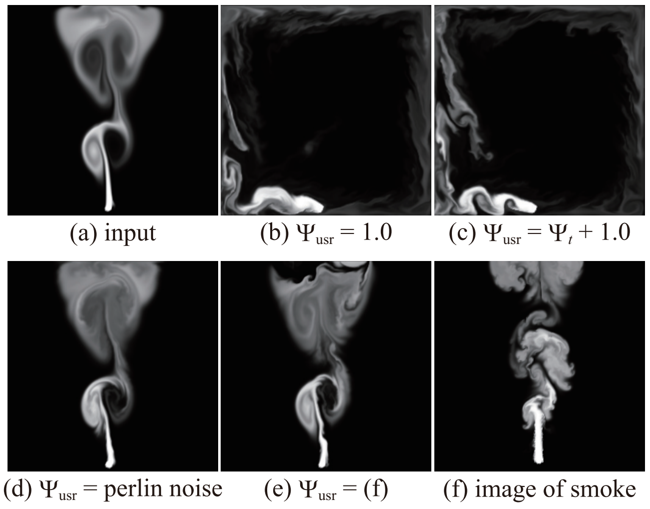

Fig. 9. 2D experimental results regarding $\mathbf{\Psi}_{\text{usr}}$.

6 CONCLUSIONS

We have proposed a guiding method that works with *stream functions*, and introduced a simple *scaling* function to formulate a minimization problem, which significantly reduces our problem size compared to previous guiding methods. Input velocities are converted into stream functions, and the final velocity is guaranteed to be incompressible.

A limitation of our method is that the boundary of the simulation space is restricted, by construction, to solid boundaries. Treating other boundary conditions is a challenging problem and left for our future work. We also plan to apply our method to other fluid phenomena, such as, fire, liquids, and clouds.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their constructive comments to improve our paper. This work was supported by JSPS KAKENHI Grant Numbers JP20K19945 and JP20H05954.

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. Bhatia, G. Norgard, V. Pascucci, and P. Bremer. 2013. The helmholtz-hodge decomposition - a survey. *IEEE Transactions on Visualization and Computer Graphics* 19, 8 (2013), 1386–1404.

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

Robert Bridson, Jim Hourihan, and Marcus Nordenstam. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics* 26, 3 (2007), Article 46.

M. Chu and N. Thuerey. 2017. Data-Driven Synthesis of Smoke Flows with CNN-based Feature Descriptors. *ACM Transactions on Graphics* 36, 4 (2017), Article 14.

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.

Zahra Forootaninia and Rahul Narain. 2020. Frequency-domain smoke guiding. *ACM Trans. Graph.* 39, 6, Article 172 (Dec. 2020).

James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. 2014. From Capture to Simulation: Connecting Forward and Inverse Problems in Fluids. *ACM Trans. Graph.* 33, 4, Article 139 (July 2014), 11 pages.

T. Inglis, M.-L. Eckert, J. Gregson, and N. Thuerey. 2017. Primal-Dual Optimization for Fluids. *Computer Graphics Forum* 36, 8 (2017), 354–368.

Doyub Kim. 2017. *Fluid Engine Development.* CRC Press.

Theodore Kim, Nils Thurey, Doug James, and Markus Gross. 2008. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics* 27, 3 (2008), Article 3.

R. Narain, J. Sewall, M. Carlson, and M. C. Lin. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Transactions on Graphics* 27, 5 (2008), Article 166.

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.

Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. 2018. Example-based Turbulence Style Transfer. *ACM Trans. Graph.* 37, 4, Article 84 (2018).

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.

H. Schechter and R. Bridson. 2008. Evolving sub-grid turbulence for smoke animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 1–7.

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

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.

Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. 2003. Discrete multiscale vector field decomposition. *ACM Transactions on Graphics* 22, 3 (2003), 445–452.