Example-based Turbulence Style Transfer

SYUHEI SATO, DWANGO Co., Ltd., Dwango CG Research
YOSHINORI DOBASHI, Hokkaido University and Dwango CG Research
THEODORE KIM, Pixar Animation Studios
TOMOYUKI NISHITA, Dwango CG Research and Hiroshima Shudo University

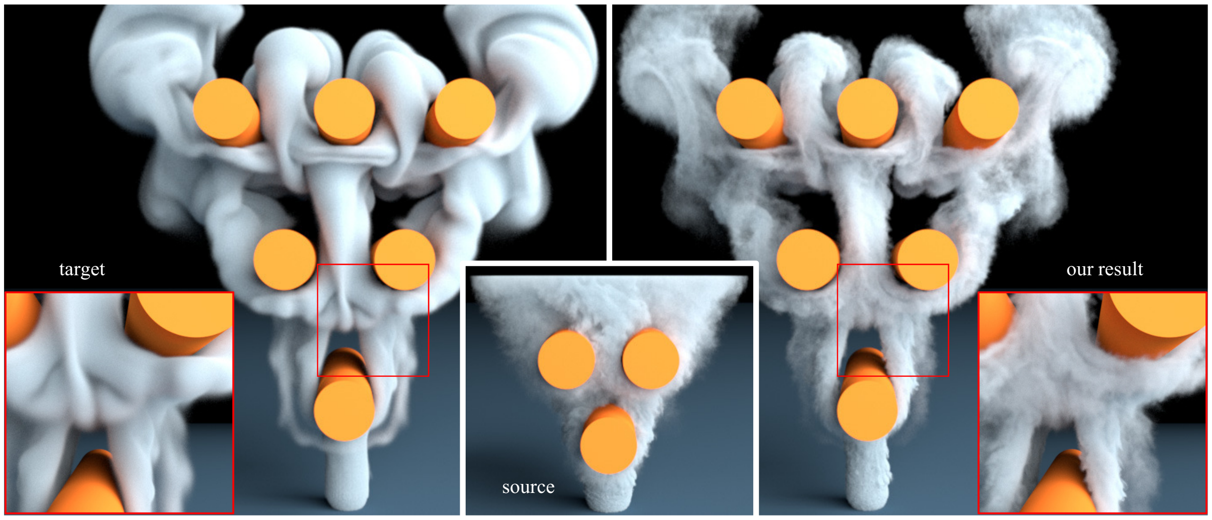

Fig. 1. Example of smoke interacting with obstacles. On the left is the original, low-resolution smoke ("target"). On the right, we transferred high-resolution turbulent motion from the middle image onto the low-resolution original. Closeups of the red square regions are shown in the insets.

Generating realistic fluid simulations remains computationally expensive, and animators can expend enormous effort trying to achieve a desired motion. To reduce such costs, several methods have been developed in which high-resolution turbulence is synthesized as a post process. Since global motion can then be obtained using a fast, low-resolution simulation, less effort is needed to create a realistic animation with the desired behavior. While much research has focused on accelerating the low-resolution simulation, the problem controlling the behavior of the turbulent, high-resolution motion has received little attention. In this paper, we show that *style transfer* methods from image editing can be adapted to transfer the turbulent style of an existing fluid simulation onto a new one. We do this by extending example-based image synthesis methods to handle velocity fields using a combination of patch-based and optimization-based texture synthesis. This approach allows us to take into account the incompressibility condition, which we have found to be a important factor during synthesis. Using our method, a user can easily and intuitively create high-resolution fluid animations that have a desired turbulent motion.

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

Additional Key Words and Phrases: fluid simulation, reusing existing fluid animations, incompressibility, texture synthesis, patch-based synthesis

ACM Reference Format:
Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. 2018. Example-based Turbulence Style Transfer. *ACM Trans. Graph.* 37, 4, Article 84 (August 2018), 9 pages. https://doi.org/10.1145/3197517.3201398

Authors' addresses: Syuhei Sato, DWANGO Co., Ltd., Dwango CG Research, syuhei_sato@dwango.co.jp; Yoshinori Dobashi, Hokkaido University, Dwango CG Research, doba@ime.ist.hokudai.ac.jp; Theodore Kim, Pixar Animation Studios, tkim@pixar.com; Tomoyuki Nishita, Dwango CG Research, Hiroshima Shudo University, nishita@shudo-u.ac.jp.

1 INTRODUCTION

Physically-based fluid simulation is used extensively in many production environments, such as movies. However, they remain computationally expensive, and animators must tediously repeat simulations multiple times in order to find parameter settings that produce a desired motion. In order to accelerate these design iterations, post-processing approaches have been proposed that add plausible turbulence [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008] and utilize guide-based formulations [Nielsen and Christensen 2010; Nielsen et al. 2009].

These methods all follow the same workflow: the overall motion is first efficiently authored at a low-resolution, and small-scale, high-frequency details are then synthesized as a post-process. While much effort has been invested in accelerating low-resolution simulations, surprisingly little research exists that addresses the problem of designing the small-scale detail. Most existing methods simply use existing procedural noise functions [Lagae et al. 2010].

However, the appearance of the final fluid motion can be considerably different depending on the representation of the small-scale details, even while the overall input motion remains the same. Unfortunately, since the small-scale details are generated entirely during the post-process, its overall appearance will not be clear to the user until after this lengthier process has completed.

In this paper, we present *turbulence style transfer*, an approach that allows the small-scale details of high-resolution simulation to be added to a low-resolution flow. While previous approaches allowed the user to design a low-resolution flow without having to consider the high-resolution features, our approach enables the reverse. The user can design a high-resolution flow with the desired turbulence characteristics once, and then repeatedly transfer these details onto low-resolution flows. Fig. 1 shows an example of smoke designed by our method.

We enable this new transfer-based approach by extending existing example-based image synthesis approaches [Kwatra et al. 2005] to handle vector fields. Our approach proceeds in two stages (see Fig. 2). First, we use patch-based synthesis to transfer high-frequency turbulent motion in a way that preserves the low-frequency content. Second, we apply optimization-based texture synthesis to resolve discontinuities between patch boundaries in a way that incorporates incompressibility. Full incompressibility is not necessary to achieve a plausible result, so we provide a user parameter that controls its amount.

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.

Several post-processing approaches for synthesizing turbulence have been proposed [Thuerey et al. 2013]. Kim et al. [2008], Narain et al. [2008] and Schechter and Bridson [2008] synthesized detailed turbulent motion on top of low-resolution simulations using a variety of procedural noise functions. Nielsen et al. [2009] took a variational approach and controlled a high-resolution fluid simulation using a user-designed low-resolution simulation. Later work accelerated the method [Nielsen and Christensen 2010]. Precomputation was explored by Pfaff et al. [2009], who synthesized object-induced turbulence by precomputing artificial boundary layer data. In contrast, Sato et al. [2012] used a precomputed database of high-resolution 2D velocity fields to synthesize high-resolution velocity fields in 3D.

Previous synthetic turbulence models use procedural methods such as Curl Noise [Bridson et al. 2007] as their high-resolution model. Chu and Thuerey [2017] instead learned a more sophisticated model by training a convolutional neural network (CNN). In contrast, we allow a user to design a velocity field that contains the turbulence characteristics they desire using whatever existing tools they prefer, and transfer the details of this design onto a new simulation. With CNN training, the learning algorithm must be re-run for a new high-resolution flow is introduced. While Chu and Thuerey [2017] do not report any timing for this stage, machine learning training times are generally quite long. Our method does not require any training; the user simply re-runs the source simulation with different parameters.

Many optimization-based methods have been proposed in texture synthesis [Barnes and Zhang 2017; Wei et al. 2009]. One popular method is that of Kwatra et al. [2005], where by successively performing a nearest neighbor search and solving a linear system, a texture that is sufficiently similar to an exemplar is synthesized. Kwatra et al. [2007] and Bargteil [2006] proposed two similar methods for synthesizing a texture on a water surface, and Narain et al. [2007] extended the method to add feature-guided details to the surface from example images. When images of water, foam or lava, are given to this system, an appropriate image is automatically selected according to the local features of the water surface. We instead focus on smoke on a grid, so these liquid surface-based methods are orthogonal to our approach. Jamriska et al. [2015] proposed an appearance transfer method for 2D fluid animations based on Kwatra et al. [2005], where the boundary and interior of the fluid are specified with an alpha mask. However, this method is intended for 2D animation and does not take into account incompressibility, so it is again orthogonal to the current work.

Ma et al. [2009] synthesized the small-scale detail in a motion field from an example image using the method of Kwatra et al. [2005]. The motion field is then added to a low-resolution motion field, but is uniform over the entire field. The turbulence in most fluid animations is heterogeneous in both space and scale, so this method is not suitable for the current task. We instead take into account local features of the input velocity and density fields and use a similarity measure to synthesize plausible turbulence.

The two stage Expectation-Maximization approach from Kwatra et al. [2005] has basic connections to the Alternating Direction Method of Multipliers (ADMM), which has been used for fluid reconstruction [Gregson et al. 2014], as well as primal-dual methods, which were used to accelerate the guided shapes approach of Nielsen et al. [2009] in Inglis et al. [2017]. However, none of these previous approaches applied these methods to the problem of style transfer.

3 OVERVIEW AND DEFINITIONS

In this section, we will give an overview of our algorithm, and define the symbols and terminology we will use throughout this paper. We use unbolded letters (d) to denote scalar quantities such as density, and boldface ($\mathbf{u}$) to denote vector quantities such as velocity. Additionally, we use lower case ($\mathbf{u}$ or d) to denote low-resolution grids, and upper case ($\mathbf{U}$ or D) for high-resolution grids. For purposes of exposition, we assume 2D grids.

Our goal is to transfer small-scale turbulence from a high-resolution simulation onto a low-resolution velocity field. We use texture synthesis to achieve this goal, but since these techniques do not take incompressibility into account, their direct application does not yield plausible results. Fig. 2 summarizes this process.

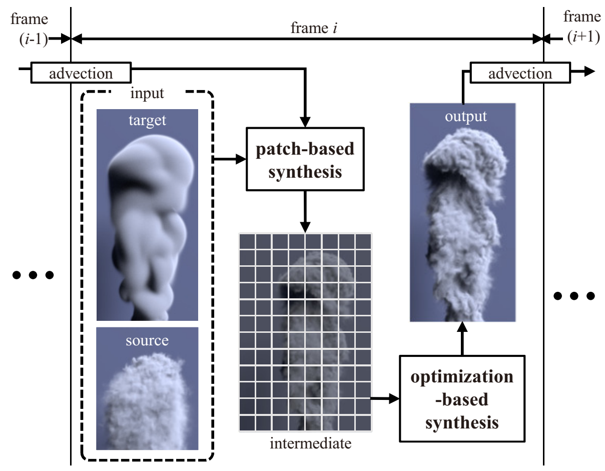

Fig. 2. Our method takes low- and high-resolution fields as input, which we denote as "target" and "source", respectively. Our patch-based synthesis stage first creates an intermediate high-resolution field. The final output is computed by applying optimization-based synthesis to this intermediate field to remove discontinuities between patches.

Table 1. Symbol definitions.

$\mathbf{u}_t, d_t$	input target velocity field, density mask
$\mathbf{U}_t, D_t$	upsampled version of $\mathbf{u}_t, d_t$
$\mathbf{U}_s, D_s$	source velocity field, density mask
$\mathbf{u}_s, d_s$	downsampled version of $\mathbf{U}_s, D_s$
$\mathbf{U}_{sh}, \mathbf{U}_{sl}$	high/low-frequency components of $\mathbf{U}_s$
$\mathbf{U}^*_h$	intermediate high-frequency components
$\mathbf{U}_h$	boundary-smoothed high-frequency components
$\hat{\mathbf{U}}_h$	advected version of $\mathbf{U}_h$
$\mathbf{U}_{final}$	final output velocity field ($= \mathbf{U}_h + \mathbf{U}_t$)
$\Phi_n(\mathbf{x}, \mathbf{f})$	$n \times n$ size patch at point $\mathbf{x}$ on field $\mathbf{f}$

We take as input low-resolution velocity ($\mathbf{u}_t$) and density mask (d_t) fields, as well as equivalent high-resolution fields ($\mathbf{U}_s$ and D_s). As indicated by the subscripts, we refer to these as the "target" and "source" fields. We assume that the number of frames in the two input simulations is the same, and if a frame number i needs to be specified, we use the indexing notation $\mathbf{u}_t(i)$.

In the first stage of our algorithm, we decompose a low-resolution velocity field into square patches, and use patch-based texture synthesis to add high-resolution details to each patch. This stage involves both a spatial and frequency-based decomposition.

The spatial decomposition is the aforementioned square patches, and we label each patch as $\Phi_n(\mathbf{x}, \mathbf{f})$. This indicates that the patch lives in field $\mathbf{f}$, is centered at point $\mathbf{x}$, and is composed of $n \times n$ grid cells. The patches are additionally defined on two spatial scales, *broad* and *narrow*, respectively on the low and high resolution grids (see Fig. 3). These patches can be defined on both the target and the source fields. The broad patches are composed of $b \times b$ grid cells, while the narrow patches contain $B \times B$ cells. For example, $\Phi_b(\mathbf{p}, \mathbf{u}_t)$ denotes a broad patch centered at point $\mathbf{p}$ on $\mathbf{u}_t$ (see Fig. 3).

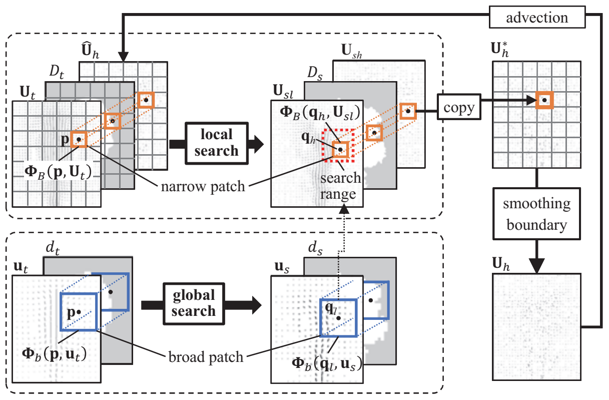

Fig. 3. A two-level search is performed, first over broad patches (blue squares) and then narrow patches (orange squares). A global search is first performed over all broad patches to find the best match, and then a narrower, local search is performed inside the best broad patch (red dotted square). The result of this stage is the intermediate velocity field $\mathbf{U}^*_h$, which is created by copying the most similar patch in $\mathbf{U}_{sh}$ to $\mathbf{U}^*_h$.

The frequency decomposition is obtained by downsampling the source velocity field, upsampling it, and then taking the difference with respect to the original. The downsampled versions of the source fields are denoted by $\mathbf{u}_s$ and d_s, and the low- and high-frequency components of the source velocity field are denoted by $\mathbf{U}_{sl}$ and $\mathbf{U}_{sh}$, respectively. Linearly upsampled versions of the target velocity and density fields are denoted $\mathbf{U}_t$ and D_t.

In the second stage of our algorithm, we take the result of the patch-based synthesis, $\mathbf{U}^*_h$, and optimize for smooth velocities across patch boundaries. This takes the form of a constrained optimization that takes into account incompressibility. We denote the boundary-smoothed result as $\mathbf{U}_h$. We use an advection version of $\mathbf{U}_h$, denoted $\hat{\mathbf{U}}_h$, to maintain temporal coherence in the patch-based synthesis stage. The final velocity field is computed as $\mathbf{U}_{final} = \mathbf{U}_h + \mathbf{U}_t$, and used to advect the density field that is shown in the final animation.

For convenience, all the symbols are summarized in Table 1.

4 PATCH-BASED TURBULENCE SYNTHESIS

In this first stage of our algorithm, we synthesize a preliminary high-resolution velocity field $\mathbf{U}^*_h$ that contains high-frequency components from the source field. This is done in a patch-based manner, so discontinuities will exist across patch boundaries that will be addressed in the next section.

We start by regularly subdividing the upsampled target field $\mathbf{U}_t$ into narrow patches of size $B \times B$ (see Fig. 3, top left). For each of these patches, the goal is then to find the patch in the source field that best matches it. The high-frequency component from this best patch will then be copied into the preliminary field $\mathbf{U}^*_h$.

A naïve approach would be to look for the best patch using an exhaustive, brute-force search. We instead elect to use a two-level algorithm that consists of a global and local stage. A global search is first performed over all the broad patches on the low resolution grids (see Fig. 3, bottom left), and once a promising patch has been found,

ALGORITHM 1: Patch-based turbulence synthesis

```
for each patch center $\mathbf{p}$ on target do
    if smoke initially appears in the patch then
        // global search
        $E_{tmp} \leftarrow \infty$
        for each patch center $\mathbf{q}$ on downsampled source do
            Compute energy: $E_l(\mathbf{p}, \mathbf{q})$ by Eq. (1)
            if $E_{tmp} > E_l(\mathbf{p}, \mathbf{q})$ then
                $E_{tmp} \leftarrow E_l(\mathbf{p}, \mathbf{q})$, $\mathbf{q}_l \leftarrow \mathbf{q}$
    else
        $\mathbf{q}_l \leftarrow \mathbf{q}_h$ in $(i-1)$-th frame
    // local search
    $E_{min} \leftarrow \infty$
    for each patch center $\mathbf{q}$ around $\mathbf{q}_l$ on source do
        Compute energy: $E_h(\mathbf{p}, \mathbf{q})$ by Eq. (2)
        if $E_{min} > E_h(\mathbf{p}, \mathbf{q})$ then
            $E_{min} \leftarrow E_h(\mathbf{p}, \mathbf{q})$, $\mathbf{q}_h \leftarrow \mathbf{q}$
    $\mathbf{\Phi}_B(\mathbf{p}, \mathbf{U}_h^*) \leftarrow \mathbf{\Phi}_B(\mathbf{q}_h, \mathbf{U}_{sh})$
```

a fine-grained local search is performed at the higher resolution. A global search over the full high-resolution data is thus avoided.

During the global search, each patch in the target field $(\mathbf{u}_t, d_t)$, must find a closest match in the downsampled, low-resolution source field $(\mathbf{u}_s, d_s)$. For each patch with center $\mathbf{p}$ in the target field, we search for the best matching position $\mathbf{q}_l$ in the source field by solving the following minimization problem.

$$\mathbf{q}_l = \arg\min_{\mathbf{q}} E_l(\mathbf{p}, \mathbf{q}),$$

where

$$E_l(\mathbf{p}, \mathbf{q}) = ||\mathbf{\Phi}_b(\mathbf{p}, \mathbf{u}_t) - \mathbf{\Phi}_b(\mathbf{q}, \mathbf{u}_s)||^2 + \alpha||\mathbf{\Phi}_b(\mathbf{p}, d_t) - \mathbf{\Phi}_b(\mathbf{q}, d_s)||^2. \quad (1)$$

Here, α is a regularization coefficient for the second term, which measures the differences in densities.

Next, we take the best broad patch found using Eq. (1) and perform a high-resolution local search along its interior to find the most similar narrow patch inside the broad patch (Fig. 3, top left). We search the source field (Fig. 3, red dotted square) for $\mathbf{q}_h$, the patch center that satisfies the following minimization:

$$\mathbf{q}_h = \arg\min_{\mathbf{q}} E_h(\mathbf{p}, \mathbf{q}),$$

where

$$\begin{aligned} E_h(\mathbf{p}, \mathbf{q}) = {} & ||\mathbf{\Phi}_B(\mathbf{p}, \mathbf{U}_t) - \mathbf{\Phi}_B(\mathbf{q}, \mathbf{U}_{sl})||^2 \\ & + \alpha||\mathbf{\Phi}_B(\mathbf{p}, D_t) - \mathbf{\Phi}_B(\mathbf{q}, D_s)||^2 \\ & + \beta||\mathbf{\Phi}_B(\mathbf{p}, \hat{\mathbf{U}}_h) - \mathbf{\Phi}_B(\mathbf{q}, \mathbf{U}_{sh})||^2. \end{aligned} \quad (2)$$

Here, α takes into account density differences in the same way as Eq. (1), and β is a regularization coefficient that adjusts the influence of previous frame. The intermediate turbulent velocity field is then synthesized by copying the high frequency component of the best patch from $\mathbf{U}_{sh}$ to the corresponding position in $\mathbf{U}_h^*$.

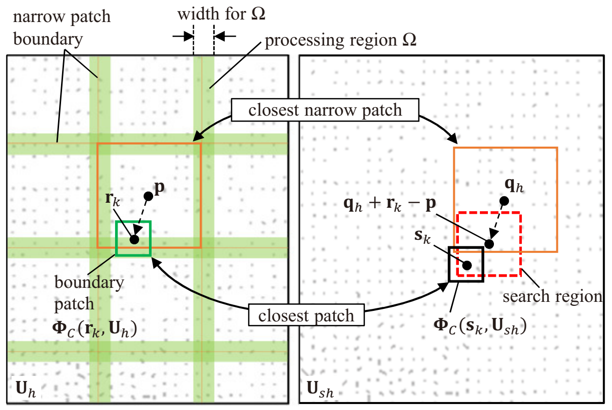

Fig. 4. Closeup around most similar narrow patches at $\mathbf{p}$ and $\mathbf{q}_h$ found in the patch-based synthesis. A processing region Ω (the green grid) and a boundary patch (the green square) are shown. The patch at $\mathbf{s}_k$ in $\mathbf{U}_{sh}$ on the right is the closest patch to the boundary patch at $\mathbf{r}_k$ in $\mathbf{U}_h$ on the left. The search region for $\mathbf{s}_k$ in the first iteration of the optimization process is defined by using the most similar narrow patch at $\mathbf{q}_h$ (the orange square on the right).

ALGORITHM 2: Smoothing velocities between patches

```
$\mathbf{U}_h^0 \leftarrow \mathbf{U}_h^*$
for $n \leftarrow 0$ to $N_{\max}$ do
    for each patch center $\mathbf{r}_k$ do
        $\mathbf{s}_k^n \leftarrow$ center of nearest neighbor of $\Phi_C(\mathbf{r}_k, \mathbf{U}_h^n)$ in $\mathbf{U}_{sh}$
    $\mathbf{U}_h^{n+1} \leftarrow \arg\min_{\mathbf{U}_h} E_b(\mathbf{U}_h, \mathbf{s}_1^n, \mathbf{s}_2^n, \cdots, \mathbf{s}_{N_\Omega}^n)$
    if $E_b$ is small enough then
        $\mathbf{U}_h \leftarrow \mathbf{U}_h^{n+1}$
        Break
```

Two factors must be taken into account to maintain temporal coherence. First, we only perform the global search when smoke initially appears in a patch. In subsequent frames, the broad patch is translated so that its center coincides with that of the best narrow patch computed at the previous frame. Second, we advect the high-frequency components from the previous $(i-1)$-th frame, $\mathbf{U}_h(i-1)$, using the upsampled target velocity field $\mathbf{U}_t(i-1)$. The resulting field is labelled $\hat{\mathbf{U}}_h$ (see Fig. 3).

Finally, we additionally accelerate the search by leveraging the density fields. The d_t and D_s field we use are not the original density fields, but indicator functions that are set to 1 when the smoke density exceeds a threshold, and 0 otherwise. The search is only performed for patches with non-zero indicator values, and thus avoids spurious computation in regions of zero density.

Algorithm 1 summarizes the patch-based synthesis stage.

5 SMOOTHING VELOCITIES BETWEEN PATCHES

In the previous section, we synthesized a high-resolution velocity field $\mathbf{U}_h^*$. However, we have ignored the boundaries between patches, so velocity discontinuities can appear along these seams.

In this section, we will propose an optimization method based on the texture synthesis approach of Kwatra et al. [2005] that addresses these discontinuities. The output will be the final velocity field $\mathbf{U}_h$.

The optimization only needs to occur along patch boundaries, so we define a lattice-structured region Ω that is the union of all the patch boundaries in $\mathbf{U}_h$ (see the green grid in Fig. 4). The width of the boundary region is specified by the user, but we found that a width of 4 worked fine in all of our experiments.

We want to make the velocity field in Ω as smooth as possible, but also preserve the high-frequency details of the source field. Thus, we propose to use vorticity to measure the similarity of the current solution and the source field. We define a $C \times C$ patch at every grid point $\mathbf{r}_k$ in Ω, where $k \in 1, 2, \cdots, N_\Omega$ and N_Ω is the total number of grid points in Ω. We call this a *boundary patch* (Fig. 4, green square). Using the boundary patches, we minimize the following energy over the Ω region in $\mathbf{U}_h$:

$$E_b(\mathbf{U}_h, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{N_\Omega}) = \sum_{k=1}^{N_\Omega} \{ ||\nabla \times \mathbf{\Phi}_C(\mathbf{r}_k, \mathbf{U}_h) - \nabla \times \mathbf{\Phi}_C(\mathbf{s}_k, \mathbf{U}_{sh})||^2 + \gamma (\nabla \cdot \mathbf{\Phi}_C(\mathbf{r}_k, \mathbf{U}_h))^2 \}. \quad (3)$$

This energy is minimized with respect to two primary variables: $\mathbf{U}_h$ and $\mathbf{s}_k$, where $k \in 1, \cdots, N_\Omega$. The patch centers $\mathbf{r}_k$ and $\mathbf{s}_k$ refer to the patches whose vorticities are most similar. The corresponding patch $\mathbf{\Phi}_C(\mathbf{r}_k, \mathbf{U}_h)$ is centered at $\mathbf{r}_k$ on the synthesized field $\mathbf{U}_h$, while $\mathbf{\Phi}_C(\mathbf{s}_k, \mathbf{U}_{sh})$ is centered at $\mathbf{s}_k$ on the high-frequency field $\mathbf{U}_{sh}$ (see Fig. 4). The first term in this energy compares the vorticity between the synthesized flow and the source flow, and the second term incorporates the incompressibility condition, which measures the divergence of the current velocity field. The γ parameter allows the weight for the incompressibility condition to be adjusted.

The energy is minimized using an Expectation-Maximization approach that is similar to Kwatra et al. [2005]. The algorithm begins by initializing $\mathbf{U}_h$ to $\mathbf{U}_h^*$. It then proceeds to alternately optimize E_b with respect to two variables: $\mathbf{s}_k$ and $\mathbf{U}_h$. In the first $\mathbf{s}_k$ optimization phase, it finds the most similar patch on $\mathbf{U}_{sh}$ for every patch centered at $\mathbf{r}_k$ (see Fig. 4). In the second phase, the algorithm minimizes E_b with respect to $\mathbf{U}_h$. This is accomplished by setting the derivative of Eq. (3) with respect to $\mathbf{U}_h$ equal to zero and by solving the resulting linear system:

$$-\nabla \times (\nabla \times \mathbf{U}_h(\mathbf{r}_k)) + \gamma \nabla (\nabla \cdot \mathbf{U}_h(\mathbf{r}_k)) = -\frac{1}{N} \sum_{l=1}^{N_\Omega} g_{kl} \nabla \times (\nabla \times \mathbf{U}_{sh}(\mathbf{s}_l)), \quad (4)$$

where $\mathbf{U}_h(\mathbf{r}_k)$ and $\mathbf{U}_{sh}(\mathbf{s}_l)$ are point-sampled velocities from $\mathbf{U}_h$ and $\mathbf{U}_{sh}$ at locations $\mathbf{r}_k$ and $\mathbf{s}_l$. The variable g_{kl} is an indicator function that is equal to one if the boundary patch at $\mathbf{r}_k$ overlaps the nearby boundary patch at $\mathbf{r}_l$, and is zero otherwise. Since we only minimize E_b over the region Ω, the linear system is relatively small. The linear system is similar to those from previous methods [Sato et al. 2016] so we use a conjugate gradient (CG) solver. After solving for $\mathbf{U}_h$, the set of closest patch centers $\mathbf{s}_k$ may also change. Hence we iteratively apply the $\mathbf{s}_k$ and $\mathbf{U}_h$ optimization phases until the change in E_b becomes sufficiently small.

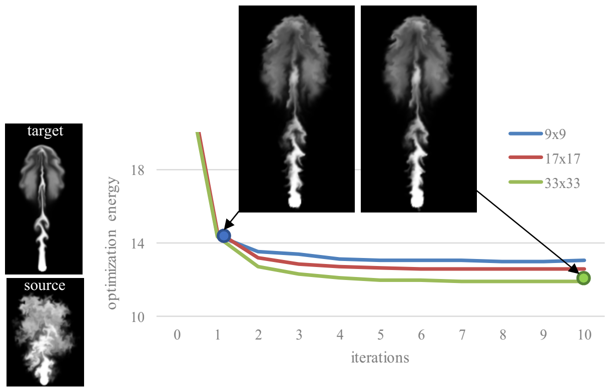

Fig. 5. Convergence of optimization energy over three different search ranges for smoothing: 9×9, 17×17, and 33×33. The images on the top show the synthesized 2D smoke with the energies corresponding to the points in the plot at the bottom, (Left: 9×9, 1 iteration) and (Right: 33×33, 10 iterations). The target and source are shown on the left.

For efficiency, we limit the search range for each $\mathbf{s}_k$ using the results of the patch-based synthesis stage. Let us assume that the center of a boundary patch, $\mathbf{r}_k$, is located within the narrow patch at $\mathbf{p}$ (see Fig. 4). The most similar patch in $\mathbf{U}_{sh}$, centered at $\mathbf{q}_h$, was already found during the patch-based synthesis process. The search is thus centered at $\mathbf{q}_h + (\mathbf{r}_k - \mathbf{p})$ (see the red dashed square in Fig. 4). After the first iteration, we update the search center to $\mathbf{s}_k$.

The computational cost of the algorithm grows as we increase the search range, so we ran experiments to determine when further error reduction becomes visually negligible. Along the top of Fig. 5, we compare smoke synthesized with two different settings. On the left is a single iteration with an 9×9 search range and on the right is ten iterations with a 33×33 search range. The size of the boundary patch is 5×5. From these experiments, we concluded that a single iteration with a 9×9 search range provided sufficient visual quality, so we used these settings for all the examples in this paper.

The fact that most error reduction occurs in the first iteration is consistent with the behavior of local/global solvers in other areas of graphics [Rabinovich et al. 2017]. The energy is almost converged after just five or six iterations, and becomes smaller when using larger search ranges.

6 ACCELERATION

Although our method can successfully synthesize small scale turbulence, the search process associated with Eq. (2) can make computation times prohibitively long. If an exhaustive search is used, our method can be slower than running a direct simulation on a high-resolution grid.

The most straightforward solution is to use a hierarchical data structure such as a kd-tree. However, our experiments yielded unattractive precomputation times for clustering and populating such a tree, and the storage costs for 3D fields become an issue.

We instead opted for a fast, simple, approximate method that requires no precomputation. Our approach leverages the fact that the velocity fields in neighboring patches are usually very similar.

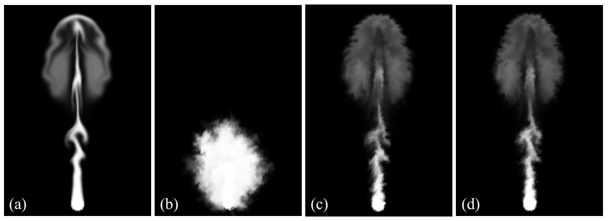

Fig. 6. Results using our adaptive search algorithm. (a) and (b) are target and source fields, respectively. (c) is synthesized by the exhaustive search and (d) is by our adaptive search. The boundary smoothing process is not applied.

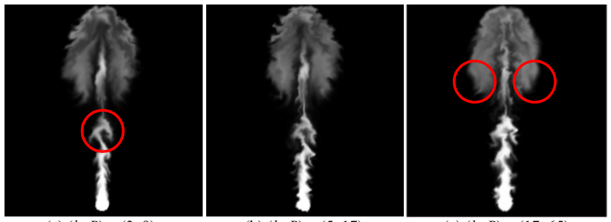

Fig. 7. Comparison of results with different patch sizes. (a) through (c) show results obtained by different broad and narrow patch sizes, b and B, indicated by the captions. The source and the target fields are the same as those in Fig. 5. The red circles indicate the regions where our method fails to transfer turbulent motion due to an inappropriate patch size.

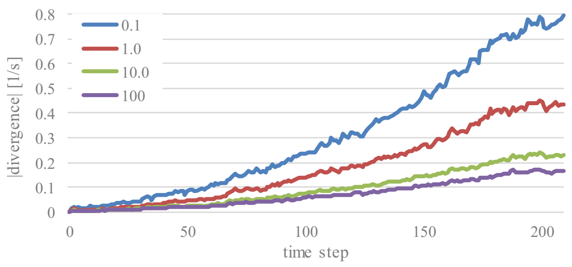

Fig. 8. Divergence observed with different $\gamma = 0.1, 1.0, 10.0$, and 100.

In lieu of an exhaustive search, we compute Eq. (2) over a subset of regularly distributed patches. The neighborhood around the best patch is then searched at more closely spaced intervals. This process repeats until the interval becomes a single voxel.

More specifically, let us assume that there are $m \times m$ patches in the search region. We first set an initial sampling interval of m_0, extract $m/m_0 \times m/m_0$ patches, and determine which patch in this subset is most similar to the query patch. Next, a new subset of patches is constructed near the most similar patch according to the sampling interval $m_1 = m_0/2$. These processes are repeated until $m_j = 1$, where j is the number of iterations. Using this adaptive approach dramatically accelerated the search process without introducing any significant artifacts. The adaptive approach was applied to both Eqs. (2) and (3).

We confirmed that this adaptive, approximate search process does not compromise the quality of the final results. In Figs. 6(a) and (b), we show a 16×24 target and 128×192 source field. We applied the adaptive search method to the local search stage, where the sizes of the narrow and broad patches, B and b, are 17 and 5, and m_0 was set to 8. Compared to the results of the exhaustive search, the average relative error of the patches found by our adaptive algorithm was 1%. The computation times of the adaptive and the exhaustive algorithms running on the CPU were 0.0125 and 0.115 seconds, respectively. Our method is 9.2× faster, and the quality in Figs. 6 (c) and (d) are comparable. In order to facilitate the comparison, the smoothing search (Eq. (3)) was deactivated for this example.

7 RESULTS

This section shows examples that demonstrate the effectiveness of our method. For all examples, the target velocity field was computed by using a Stam [1999] solver, and the turbulent source velocity field is generated using vorticity confinement [Fedkiw et al. 2001]. We used a desktop PC with an Intel Core i7-6700K CPU to compute all examples. The grid sizes, parameters, and computation times are summarized in Table 2. The computation time of our method is almost the same as or faster than that of the source simulation. In our method, more than 95% of the computation time was spent smoothing patch boundaries. If the acceleration method from §6 is not used, then patch search dominates. The rightmost column in Table 2 shows the time needed to compute full simulations at the same resolution as our final results. Running theses full-resolution simulations takes 1.5×-3.5× longer than our method (source simulation + synthesis process). The video including these examples can be found in the supplementary material.

7.1 Experiments Using 2D Flows

We investigate the behavior of our method for 2D smoke simulations. The first experiment compares the results obtained by using different sizes for broad and narrow patches, b and B. We tested three combinations of patch sizes: $(b, B) = (3, 9), (5, 17)$, and $(17, 65)$. The source and the target fields are the same as those in Fig. 5. The results are shown in Fig. 7. This experiment indicates that small patches are inappropriate for successfully transferring turbulent motion (e.g., see the region indicated by the red circle in(a)). Using patches that are too large also fails to transfer turbulent motion as shown in the red circles in (c). Based on these experiments, we decided to use $(b, B) = (5, 17)$ for creating the 3D examples in the next section.

Next, we investigated the effects of the varying parameter γ from Eq. (3), which controls the amount of incompressibility in the synthesized flow. We computed the average absolute divergence, and plotted the results for $\gamma = 0.1, 1.0, 10.0$, and 100.0 in Fig. 8. As expected, the divergence becomes smaller as γ grows larger.

7.2 Experiments Using 3D Flows

We applied our method to several practical 3D examples. The sizes of the broad and narrow patches were set to 17^3 and 5^3 in all the 3D examples. All the images in this section were rendered by the Mitsuba renderer [Jakob 2010]

Table 2. Simulation statistics. The rightmost column indicates the computation times for full simulations using the same resolution as our final results.

Figure	grid size			parameters			computation time/frame [sec]			
	target	source	final	α	β	γ	target	source	synthesis	full
Fig. 7(a)	16×24	128×192	192×256	0.001	0.5	2.5	0.001	0.31	0.083	0.83
Fig. 7(b)	16×24	128×192	192×256	0.001	0.5	2.5	0.001	0.31	0.057	0.83
Fig. 7(c)	16×24	128×192	192×256	0.001	0.5	2.5	0.001	0.31	0.039	0.83
Fig. 9(b)	$24 \times 24 \times 32$	$192 \times 192 \times 128$	$192 \times 192 \times 256$	0.001	0.5	2.5	0.03	56	51	166
Fig. 9(c)	$24 \times 24 \times 32$	$128 \times 128 \times 192$	$192 \times 192 \times 256$	0.0003	0.5	2.5	0.03	34	50	166
Fig. 11(a)	$32 \times 32 \times 48$	$256 \times 256 \times 256$	$256 \times 256 \times 384$	0.0005	0.5	2.5	0.11	260	169	615
Fig. 11(b)	$48 \times 32 \times 32$	$256 \times 256 \times 256$	$384 \times 256 \times 256$	0.0005	0.5	2.5	0.11	260	163	615
Fig. 1	$64 \times 32 \times 64$	$256 \times 256 \times 256$	$512 \times 256 \times 512$	0.0005	0.5	2.5	0.45	260	338	2170

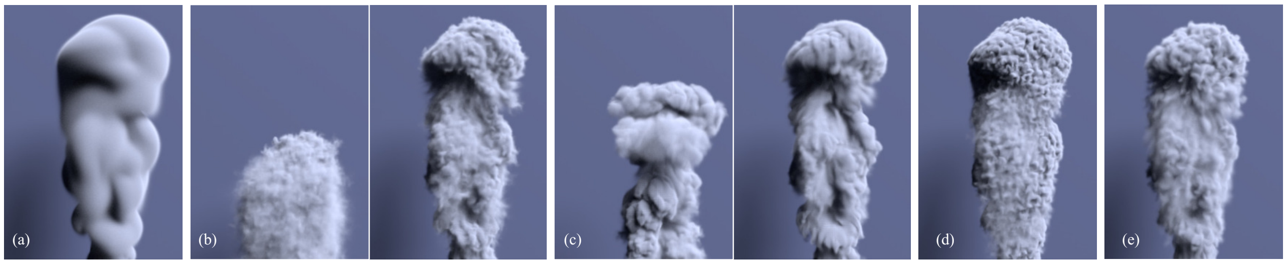

Fig. 9. Example of rising smoke. Image (a) shows a target low resolution smoke. The images on the right in (b) and (c) show our results created by transferring high resolution turbulent motions in the left images to the low resolution target smoke. Images (d) and (e) show results obtained by using Wavelet Turbulence and the method of Pfaff et al. [2009].

The first example shows rising smoke synthesized using source fields with different turbulent motions (Fig. 9). We prepared two source fields with different grid sizes for Fig. 9(b) and (c) (see Table 2). The smoke is generated by placing a spherical smoke source at the bottom-center of the simulation domain. The radius of the smoke source for the target is 2× the grid interval. The radii for the source fields in (b) and (c) are 10× and 5×, respectively. These settings were chosen experimentally. Within the smoke source, a random vertical upward velocity and a constant density were specified. Our method successfully transferred the turbulent motion from the source fields to the target field, resulting in different small scale details. Figs. 9(d) and (e) show results of using Wavelet Turbulence [Kim et al. 2008] and the Artificial Boundary Layer (ABL) method of Pfaff et al. [2009], respectively. Wavelet Turbulence successfully adds small-scale details, but the results look noisy. ABL produces better results, but for both methods, parameter tweaking is needed to obtain plausible results. Our results are more realistic because the small-scale details have been transferred from a high-resolution simulation.

In the second example, shown in Fig. 1, we applied our method to a scene where smoke interacts with multiple cylinders. The target is shown in the left. In Fig. 1, we prepared a source field by simulating smoke interacting with only three cylinders (bottom middle), on a grid with half the size of the final simulation. The high-resolution turbulent smoke synthesized by our method is shown in the right. Even though the source field only has half of the grid size of the final result, the small scale details are transferred effectively.

Fig. 10 investigates the ability of our method to use different source and target fields in the same scene as Fig. 1. Fig. 10(a) uses the same source field as Fig. 1 but vorticity confinement has been changed to reduce turbulence. In Fig. 10(b), the source field shown in Fig. 9(c) is directly used, so no cylinders exist in the source field. In Fig. 10(c), the cylinders in the target were replaced with the large flat slab shown in the top left corner. These experiments show that our method can transfer high-resolution details even if the source and the target simulations are fairly different. Finally, in Fig. 10(d), we simulated smoke with the same resolution as our result in Fig. 1. Although realistic high-resolution details are obtained, the overall motion is completely different from the target simulation, and the computation time is much longer (see Table 2).

Finally, we applied our method to more complicated scenarios where smoke interacts with moving objects (Fig. 11). In the target low-resolution simulation in Fig. 11(a), a sphere moves up and down *dynamically*. However, the source field was created by simulating smoke interacting with a *static* sphere. In Fig. 11(b), the cylinders move from left to right, while the same static source animation from Fig. 1 is used. Our method synthesizes plausible high-resolution flows, even in these difficult situations.

8 DISCUSSION

Our method combines patch-based and optimization-based texture synthesis. When using such techniques, selecting a good patch size is an important factor in determining the quality of the final result. Although we have not encountered a case where choosing an appropriate patch size is difficult, it is possible that such scenarios could arise. In such situations, the issue could be resolved by incorporating a hierarchical texture synthesis approach.

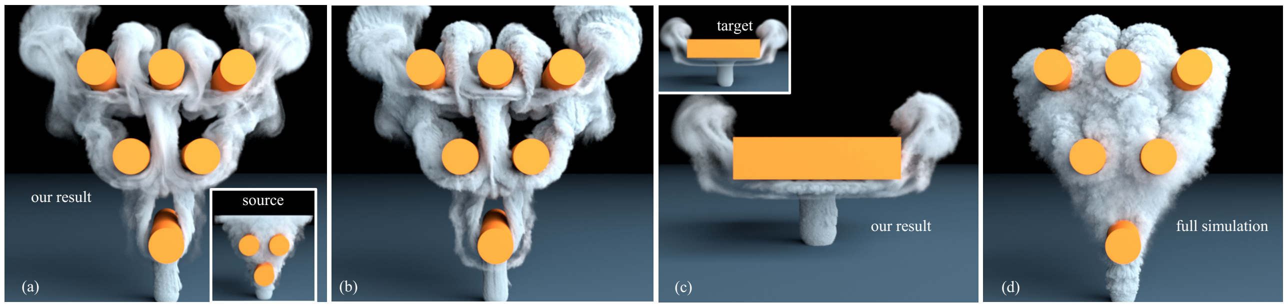

Fig. 10. Scenario from Fig. 1, but with different source/target combinations. The source for (a) is shown at the bottom right corner, and created by reducing the turbulence from Fig. 1. Image (b) uses the source shown in Fig. 9(c). In image (c), the target is replaced with the large flat slab shown in the upper left corner. The source is the same as Fig. 1. Image (d) is the result of a direct simulation, where the resolution is the same as our result from (a).

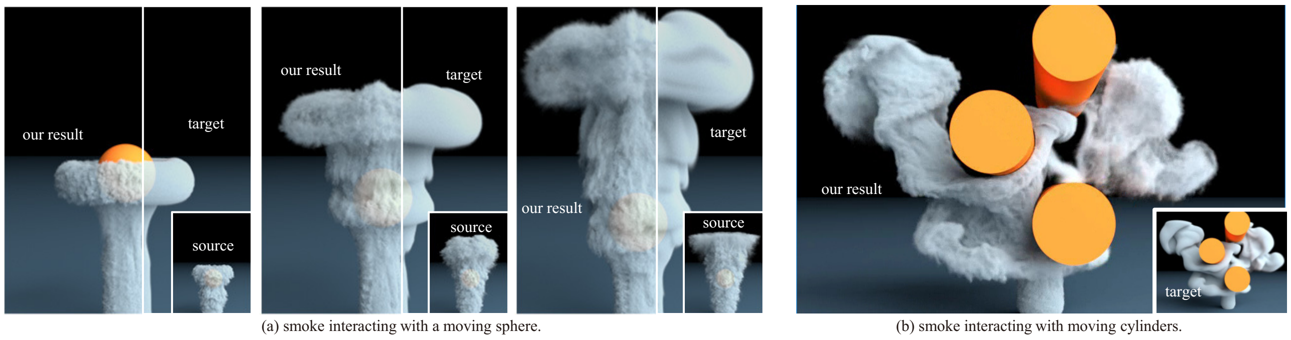
our result
target
source
our result
target
source
target
our result
source
(a) smoke interacting with a moving sphere.
our result
target
(b) smoke interacting with moving cylinders.

Fig. 11. Smoke interacting with moving obstacles. The target low-resolution smoke interacts with *dynamic* obstacles. The results are created by transferring high-resolution turbulent motions smoke interacting with *static* obstacles (see insets). In image (a), a sphere moves up and down, and in (b), cylinders move left and right. In (b), the source field is the same as Fig. 1.

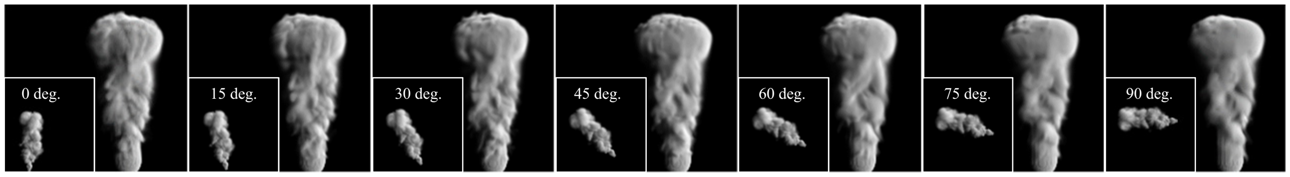

Fig. 12. Different source velocity fields with different buoyancy directions. The source fields are shown in the insets, and the buoyancy directions are listed as well. The target field is the same as Fig. 9(a).

One limitation is that we assume the source animation is at least as long as the target animation. There are many possible solutions to this problem, including the infinite "Video Textures" approach from Schödl et al. [2000]. No new component needs to be needed to address this limitation.

One possible alternative that we discarded was the direct application of the texture synthesis technique of Kwatra et al. [2005] to the velocity field. We found that the computational cost was very large, particularly for 3D cases, and the results were not visually convincing. The entire simulation domain would suffer from excessive overuse of low-frequency patches, similar to the observations in Jamriska et al. [2015]. We instead only apply this technique across boundary patches, so the artifacts are minimal.

If the gross motion of the target and source velocity fields differ greatly, i.e. the lowest frequency components are totally mismatched, our method can produce unintuitive results. Figs. 12 through 14 explore the sensitivity of our method to these differences.

Fig. 12, shows seven source velocity fields with buoyancy directions that differ by 15 degrees. The target velocity field is the same as Fig. 9. Our method begins to fail when the buoyancy direction deviates from the purely vertical direction by over 30 degrees.

Next, in Fig. 13, two target velocity fields are generated with small (left) and large (right) horizontal velocities, and the source is the

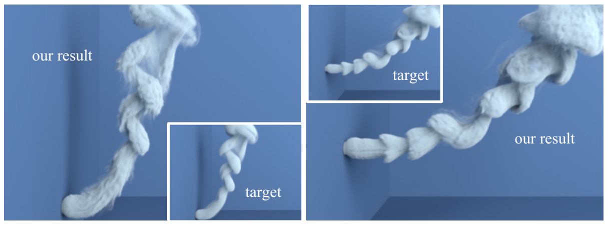

Fig. 13. Experiments using target fields with small (left) and large (right) initial horizontal velocities. The source field is the same as Fig. 9(c). The synthesis quality can degrade if the horizontal velocity becomes very large.

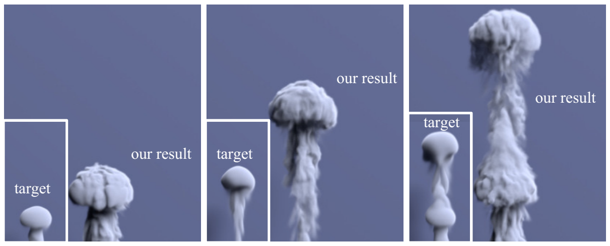

Fig. 14. Experiments in temporal similarity. The inset images show the target smoke simulation, and the source field is the same as Fig. 9(c). Even though smoke is injected intermittently in the target simulation, turbulent detail is transferred successfully.

same as Fig. 9(c). The turbulent detail is successfully transferred when the initial velocity is small, but is less successful for large velocities. This limitation could be addressed by making our method rotation- and translation-invariant, and robust to normalized flows.

Finally, we investigate the necessity of temporal similarity in Fig. 14. In the source, smoke is added continuously, while in the target, smoke is added intermittently. We use the same source as Fig. 9(c), and our method successfully transfers the turbulent detail.

9 CONCLUSIONS

We have proposed a method for synthesizing turbulent motion by reusing existing velocity fields in an incompressibility-aware manner. Our method used patches to synthesize turbulent motion that preserves the overall character and structure of the flow, and uses optimization-based texture synthesis to remove artifacts along the boundary of each patch. In the future, we want to explore accelerating our method by implementing the whole process on the GPU. Our preliminary studies indicate that it can accelerate the plume example by up to 30×. We also plan to extend our method to other types of fluid, such as fire, clouds, and water.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Hiroyuki Ochiai at Kyushu University for his valuable comments. The authors would also like to thank the reviewers for their constructive comments to improve our paper. This work was supported by JSPS KAKENHI Grant Numbers JP15H05924 and JP15K21742.

REFERENCES

Adam Wade Bargteil. 2006. *Tracking and texturing liquid surfaces.* Ph.D. Dissertation. University of California, Berkeley.

Connelly Barnes and Fang-Lue Zhang. 2017. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media* 3, 1 (2017), 3–20.

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. Fedkiw, J. Stam, and H. W. Jansen. 2001. Visual Simulation of Smoke. In *Proceedings of ACM SIGGRAPH 2001.* 15–22.

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.

Wenzel Jakob. 2010. Mitsuba renderer. (2010). http://www.mitsuba-renderer.org.

O. Jamriska, J. Fiser, P. Asente, J. Lu, E. Shechtman, and D. Sykora. 2015. LazyFluids: appearance transfer for fluid animations. *ACM Transactions on Graphics* 34, 4 (2015), Article 92.

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.

Vivek Kwatra, David Adalsteinsson, Theodore Kim, Nipun Kwatra, Mark Carlson, and Ming C. Lin. 2007. Texturing fluids. *IEEE Transactions on Visualization and Computer Graphics* 13, 5 (2007), 939–952.

Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics* 24, 3 (2005), 795–802.

Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. 2010. A survey of procedural noise functions. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 2579–2600.

C. Ma, L. Wei, B. Guo, and K. Zhou. 2009. Motion field texture synthesis. *ACM Transactions on Graphics* 28, 5 (2009), Article 110.

Rahul Narain, Vivek Kwatra, Huai-Ping Lee, Theodore Kim, Mark Carlson, and Ming C. Lin. 2007. Feature-guided dynamic texture synthesis on continuous flows. In *Proceedings of the 18th Eurographics conference on Rendering Techniques.* 361–370.

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.

T. Pfaff, N. Thuerey, A. Selle, and M. Gross. 2009. Synthetic turbulence using artificial boundary layers. *ACM Transactions on Graphics* 28, 5 (2009), Article 121.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2, Article 16 (April 2017), 16 pages.

S. Sato, Y. Dobashi, and T. Nishita. 2016. A combining method of fluid animations by interpolating flow fields. In *Proceedings of SIGGRAPH Asia 2016 Technical Briefs.* Article 4.

S. Sato, T. Morita, Y. Dobashi, , and T. Yamamoto. 2012. A data-driven approach for synthesizing high-resolution animation of fire. In *Proceedings of the Digital Production Symposium 2012.* 37–42.

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.

Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. 2000. Video Textures. In *Proceedings of ACM SIGGRAPH 2000.* 489–498.

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

Nils Thuerey, Theodore Kim, and Tobias Pfaff. 2013. Turbulent Fluids. In *Proceedings of SIGGRAPH '13 ACM SIGGRAPH 2013 Courses.* Article No. 6.

Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. 2009. State of the Art in Example-based Texture Synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR.* Eurographics Association. http://www-sop.inria.fr/reves/Basilic/2009/WLKT09