LazyFluids: Appearance Transfer for Fluid Animations

Ondřej Jamriška[1*] Jakub Fišer[1] Paul Asente[2] Jingwan Lu[2] Eli Shechtman[2] Daniel Sýkora[1]
[1]CTU in Prague, FEE [2]Adobe Research

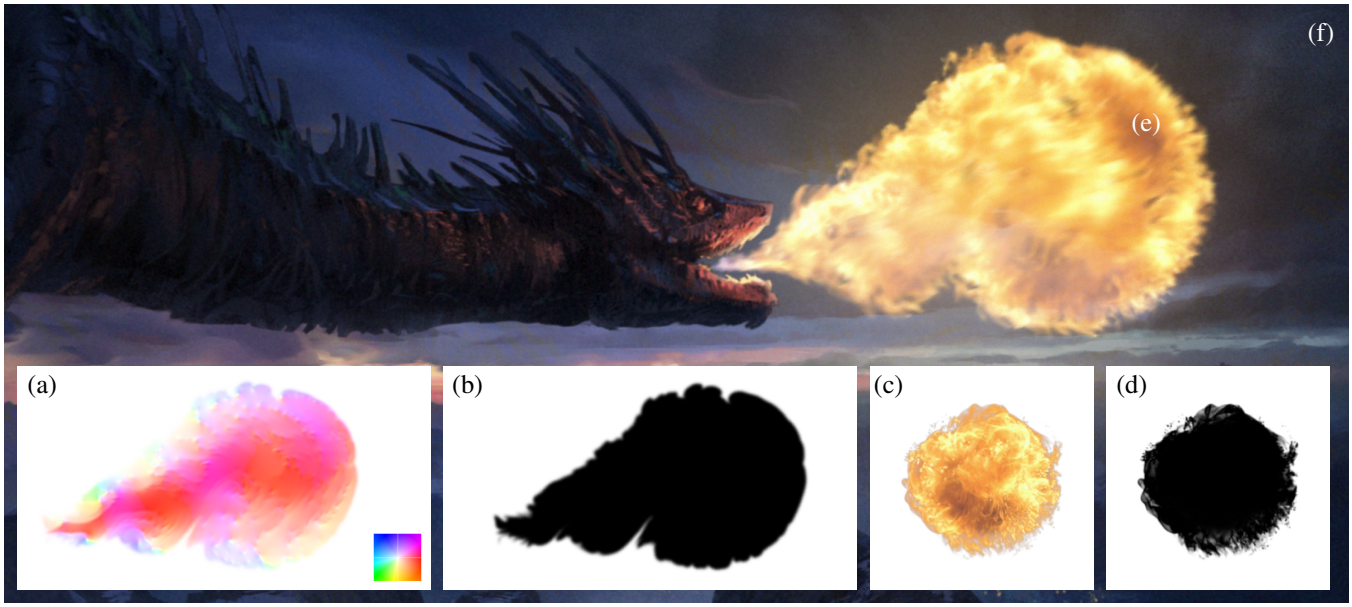


Figure 1: *LazyFluids in action—an artist first designs a target fluid animation that consists of a sequence of motion fields (a) and alpha masks (b), and then selects a video exemplar of a fluid element with desired appearance (c) and alpha mask (d). Finally, our algorithm transfers appearance of the exemplar to the target animation while respecting its motion properties and boundary effects (e). The resulting animation can then be used as a part of a more complex composition (f). All alpha masks in the paper are visualised in a way that fully opaque pixels are black and fully transparent are white. Dragon painting © Jakub Javora.*

Abstract

In this paper we present a novel approach to appearance transfer for fluid animations based on flow-guided texture synthesis. In contrast to common practice where pre-captured sets of fluid elements are combined in order to achieve desired motion and look, we bring the possibility of fine-tuning motion properties in advance using CG techniques, and then transferring the desired look from a selected appearance exemplar. We demonstrate that such a practical workflow cannot be simply implemented using current state-of-the-art techniques, analyze what the main obstacles are, and propose a solution to resolve them. In addition, we extend the algorithm to allow for synthesis with rich boundary effects and video exemplars. Finally, we present numerous results that demonstrate the versatility of the proposed approach.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations I.3.4 [Computer Graphics]: Graphics Utilities—Graphics editors I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: texture synthesis, flow-guided, example-based, temporal coherency, fluid simulation

*e-mail:jamriond@fel.cvut.cz

1 Introduction

Special effects based on fluid elements are ubiquitous in current digital movie production. To achieve a desired look, an artist typically makes a composition out of pre-captured videos of real fluids with a desired appearance. A key limitation here is that the motion properties of these videos remain fixed. When finer control is needed the artist has to resort to full fluid simulation followed by an advanced rendering algorithm. In this scenario, however, limited resolution and the complexity of material properties, lighting, or other parameters may hinder delivering the desired visual look.

We would like to offer artists a more practical workflow that can narrow the gap between appearance and motion controllability:

1. Quickly design the target animation using 2D CG techniques (e.g., a real-time fluid simulator [Stam 1999] or particle system [Reeves 1983]; see Fig. 1a,b).

2. Pick a photo or a video sequence containing the desired look (the source exemplar; see Fig. 1c).

3. Add an alpha channel to the source exemplar to distinguish between interior and boundary samples (see Fig. 1d).

4. Run an example-based synthesis algorithm to transfer appearance from the source exemplar to the target fluid animation (see Fig. 1e).

Although such example-based workflow can considerably simplify the creation of fluid elements, we found that the current state of the art in flow-guided appearance transfer [Neyret 2003; Kwatra et al. 2005; Bénard et al. 2013; Browning et al. 2014] does not solve (4), either producing disturbing temporal artifacts or failing to reproduce visual characteristics of real fluid elements.

In this paper we analyze the source of failure in methods originating from Kwatra et al. [2005] and formulate a novel optimization method that addresses it. We extend the formulation to use video exemplars and support rich boundary effects, which are crucial for compelling appearance transfer. Finally, we compare our results with the current state-of-the-art and demonstrate various realistic use cases that confirm the practical utility of our approach.

2 Related Work

Texture advection is a common approach to appearance transfer for fluid animations. This technique was pioneered by Max and Becker [1992] and later extended by others [Neyret 2003; Bousseau et al. 2007; Yu et al. 2011]. Although this simple yet effective solution produces impressive results, its key limitation is that it suffers from notable texture distortion that needs to be alleviated by blending with a new, undistorted texture source. This typically leads to disturbing ghosting artifacts. Another disadvantage is that larger texture exemplars are required to cover longer motions. In situations when the length of motion is not known *a priori*, procedurally generated textures [Perlin 1985] or blending multiple textures can alleviate this limitation.

A different approach to appearance transfer, requiring only small exemplars, was presented by Kwatra et al. [2005] and later improved by Lefebvre et al. [2006]. The method has also been extended to work on arbitrary surfaces [Han et al. 2006; Bargteil et al. 2006; Kwatra et al. 2007; Narain et al. 2007]. A great advantage of this technique is that the amount of texture distortion is effectively controlled by a texture synthesis algorithm whose aim is to match the target appearance with the source texture. Although these techniques achieve compelling results on a carefully selected set of sources we found they often fail on exemplars of real fluid elements, generating excessive repetition or flat areas.

Bhat et al. [2004] proposed a different flow-based video synthesis technique that can be classified as a middle ground between texture advection and synthesis. It uses a set of textured particles moving along user-specified flow-lines and leverages video textures [Schödl et al. 2000; Kwatra et al. 2003] to generate infinite or looped sequences from a short video exemplar. The technique can produce compelling results; however, it supports only simple flow fields and requires a suitable video source of the appropriate fluid element.

Our approach bears some resemblance to regenerative morphing techniques [Shechtman et al. 2010; Darabi et al. 2012] that produce visually interesting transitions between dissimilar images. They were recently applied to stylization of fluid simulations [Browning et al. 2014] with compelling results. Nevertheless, a key drawback is that the user must prepare a set of keyframes that roughly match the appearance of the target flow at selected time steps. When synthesizing transitions between these keyframes, these techniques can produce temporal artifacts such as ghosting and pulsation, which break the smoothness of the resulting animation.

Recently, Bénard et al. [2013] proposed an extension of image analogies [Hertzmann et al. 2001] to synthesize impressive stylized animations guided by a synthetic motion field. Although their approach can be applied in our scenario, we found that it tends to lose high frequency details (both in the interior and at the boundaries) and produces distinct temporal artifacts such as popping and drifting. These are perceived as natural in the context of stylization but feel disturbing when applied to smooth fluid animations.

3 Problem Formulation

There are two inputs to our method (see Figures 10 and 11):

- An exemplar of a fluid element Z that can be a single RGBA image or a sequence of M RGBA images $(Z^t)_{t=1}^M$.
- A sequence of N target alpha masks $(X_a^t)_{t=1}^N$ (gray-scale images) with corresponding 2D motion fields $(F^t)_{t=1}^N$.

Source Z represents the desired appearance, X_a captures the shape, and F captures the motion of the target fluid animation. The aim is to transfer the appearance from Z to X in a way that the resulting visual content moves along with F, is coherent in time, and respects boundary-specific effects prescribed by X_a and Z_a (the alpha channel of Z).

3.1 Analysis

Kwatra et al. [2005] addresses a simpler problem with no alpha masks (Z_a and X_a^t), treating the source Z as a single RGB image and the target X as a sequence of RGB images. For simplicity in the analysis we do the same and later we extend the formulation to handle alpha masks. In this simplified scenario the problem is formulated as a minimization of the following energy (originally proposed by Kwatra et al. [2005]):

$$E(Z, X^t, \hat{X}^{t-1}) = E_s(Z, X^t) + \lambda E_t(X^t, \hat{X}^{t-1}) \qquad (1)$$

where X^t is the currently synthesized frame and $\hat{X}^{t-1}$ is the previously synthesized frame, forward-warped using motion field F^{t-1}.

The energy (1) contains two terms:

- *Source coherence*:

$$E_s(Z, X^t) = \sum_{p \in X^t} \min_{q \in Z} ||\mathbf{x}_p^t - \mathbf{z}_q||^2 \qquad (2)$$

  This ensures the appearance of the target animation frame X^t is similar to the exemplar Z. Here $\mathbf{z}_q$ and $\mathbf{x}_p^t$ denote source and target patches centered at pixels $p \in X^t$ and $q \in Z$ respectively.

- *Temporal coherence*:

$$E_t(X^t, \hat{X}^{t-1}) = \sum_{p \in X^t} ||X^t(p) - \hat{X}^{t-1}(p)||^2 \qquad (3)$$

  This ensures the resulting fluid animation changes smoothly in time and moves along with F. Here $X^t(p)$ denotes color at a pixel $p \in X^t$.

To minimize (1) Kwatra et al. [2005] proposed a method that produces impressive results (see Fig. 2b) when used with carefully selected sources containing repetitive patterns where the size of the repeated elements is similar to that of the source patch z_p (Fig. 2a). However, even a small modification of these sources (Fig. 2c) can lead to notable degradation (Fig. 2d). This behavior becomes much

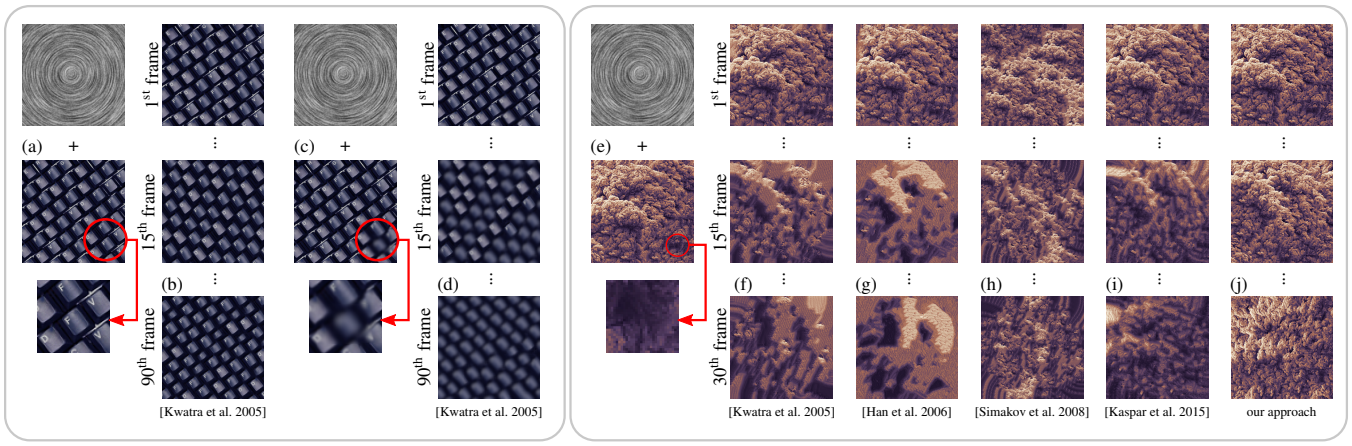


Figure 2: *Examples of gradual wash-out. Previous approach to flow-guided synthesis [Kwatra et al. 2005] produce compelling results (b) when applied to exemplars containing repetitive patterns where the size of the repeated element corresponds to the patch size used for the synthesis (a). With a source that contains areas that are comparatively smooth, like blurred areas (c) or low-contrast parts of real fluid elements (e), the results are initially good. But after several frames the resulting animation degrades into a repetitive use of the smoothest patches (d,f) because of an effect described by Newson et al. [2014]. This effect prevails even if more advanced techniques such as (g) discrete solvers [Han et al. 2006], (h) bi-directional similarity [Simakov et al. 2008], or (i) occurrence maps [Kaspar et al. 2015] are used. Our approach (j) resolves this problem thanks to the ability to enforce uniform patch usage. Smoke exemplar © Richard Roscoe.*

more apparent (Fig. 2f) when Z is an exemplar with variable content such as realistic smoke or fire elements (Fig. 2e). These exemplars are not repetitive, and they often contain areas of low contrast. We call this effect *gradual wash-out* and seek a viable strategy to avoid it.

Bargteil et al. [2006] were the first who noted that the approach of Kwatra et al. [2005] produces gradual wash-out and thought that there was a trade-off between temporal coherence and content preservation. They tried to suppress it by lowering λ in (1), i.e., reducing the influence of the temporal coherence term (3). This solution slows the degradation down, but it cannot resolve the root of the problem.

In the results of Han et al. [2006], gradual wash-out is also visible although not explicitly addressed. They mentioned a different problem that might seem to be a source of gradual wash-out—that the least-square solver used during the E-step of Kwatra et al.'s algorithm tends to produce blurring artifacts. They tried to alleviate this behavior by using a discrete solver based on k-coherence search. Although their solution can bring some improvement when synthesizing a single image, we found that it makes the gradual wash-out even worse (see Fig. 2g) and that the source of the problem is actually hidden somewhere else.

An better explanation for this erroneous behavior was recently provided by Newson et al. [2014], who show that during the nearest neighbour retrieval, textured patches are more likely to be matched with smooth ones. We observe that, in conjunction with Kwatra et al.'s algorithm, this effect leads to positive feedback that propagates smoother patches, which come to prevail. If the source Z does not contain visibly smoother patches, the algorithm tends to pick a patch or a set of patches that are as smooth as possible (e.g., areas with lower contrast) and starts to prefer them. This explains why all the methods originating from Kwatra et al. work only for exemplars consisting of repetitive patterns (see Fig. 2a) and why they fail for other exemplars (Fig. 2e).

To avoid a preference for smoother patches Newson et al. proposed using texture features, which resemble feature masks used in [Lefebvre and Hoppe 2006]. However, a fundamental issue is that it is not clear how to initialize the solution in a way that avoids excessive repetitions of texture features during the synthesis. Newson et al. used inpainting, which is not applicable in our scenario.

Wei et al. [2008] and independently Simakov et al. [2008] proposed a bi-directional similarity (BDS) measure that in addition to source coherence (2) uses a new source *completeness* term:

$$E_c(Z, X^t) = \sum_{p \in Z} \min_{q \in X^t} ||\mathbf{z}_p - \mathbf{x}_q^t||^2 \tag{4}$$

Its aim is to ensure that all source patches are represented in the synthesized output. Although this extension can bring an improvement, there is a fundamental limitation that the source must be at least as large as the target. If this is not satisfied, an optimal solution would have a few small islands of source patches with the rest filled with repetitions of the smoothest patches. This is the case in our scenario, where appearance exemplars are typically much smaller than the target (see Fig. 2h).

Kopf et al. [2007] and later Chen and Wang [2010] proposed an approach in which histogram matching of patch colors and offsets is used to bias the optimization towards a solution that penalizes excessive use of a certain subset of source patches. Recently, in concurrent work to ours, Kaspar et al. [2015] extended this technique to explicitly reject patches that have already been used more than twice the uniform usage level. Although these approaches have the potential to suppress the gradual wash-out, they can still lead to a solution with non-uniform patch usage. The key issue is that there is no mechanism to strictly enforce all source patches being used equally in the target, and thus the degradation is still visible (see Fig. 2i).

4 Our Approach

To fully avoid the preference for particular patches that leads to appearance degradation, we need to strictly preserve uniformity of patch usage. We do this by minimizing (1) subject to an additional *uniformity* constraint:

$$\sum_{p \in Z} \delta(p) = |X| \quad \text{and} \quad \delta(p) - K \in \{0, 1\} \tag{5}$$

where $\delta(p)$ counts the usage of a source patch $\mathbf{z}_p$ centered at a pixel $p \in Z$ and K is the floor of the ratio between the number of target $|X|$ and source $|Z|$ patches, i.e., $K = \lfloor |X|/|Z| \rfloor$.

To solve this new constrained optimization we draw inspiration from a concept previously proposed by Rosenberger et al. [2009] that was originally used to optimize BDS in the context of shape synthesis. There are algorithms such as Simakov et al. [2008] that can achieve better BDS. However, in our scenario the aim is not to optimize BDS but to perform synthesis in a way that the uniformity constraint (5) is satisfied. For this goal the concept proposed by Rosenberger et al. is more suitable as it provides a mechanism to satisfy uniform patch usage.

In the following sections we first demonstrate how to apply the concept of Rosenberger et al. [2009] in our scenario (Section 4.1). Then we extend it to enable rich boundary effects (Section 4.2) and temporal coherence (Section 4.3). Finally, we propose a joint formulation (Section 4.4) which encompasses all mentioned features into one optimization problem and provide two additional improvements (Section 4.5).

4.1 Nearest-neighbour Field

We made a key modification to Kwatra et al.'s [2005] algorithm to enforce uniform patch usage. We changed how the nearest-neighbour field (NNF) is computed during each iteration of the algorithm. Similarly to Rosenberger et al. [2009] we reverse the direction of NNF retrieval (cf. Fig. 3), i.e., for each source patch $\mathbf{z}_p$, $p \in Z$ we find a target patch $\mathbf{x}_q$ that has minimal distance: $D(\mathbf{z}_p, \mathbf{x}_q) = ||\mathbf{z}_p - \mathbf{x}_q||^2$. Since we do this search independently, it can happen that two source patches can identify the same target patch as their nearest neighbour. We resolve this collision by keeping the correspondence with the smaller patch distance. Moreover, since the number of patches in Z is usually smaller than the number in X we need to repeat the NNF retrieval until all patches in X have been assigned their counterparts in Z (see Fig. 3). To make sure every patch from Z is used equally in X we:

1. use a counter c_p that is initially set to 0 and then gradually incremented whenever $\mathbf{z}_p$ is assigned to $\mathbf{x}_q$ (the white numbers inside circles in Fig. 3).
2. perform nearest neighbour retrieval only between patches $\mathbf{z}_p$ with $c_p < K$ and yet unassigned patches in X (the empty circles in Fig. 3)

When $|X|$ is not divisible by $|Z|$, i.e., when there is a non-zero remainder $R = |X| \bmod |Z|$, the situation becomes more complex. Rosenberger et al. [2009] proposed randomly picking R patches from Z to even up R; however, this random pick may bias the solution towards patches that unnecessarily increase the overall energy (1). We instead propose a better solution that lets all patches equally participate during the NNF retrieval phase.

During the repeated retrieval we ease the original limitation that only patches with $c_p < K$ can be considered for assignment and also allow patches with $c_p = K$ since some of them need to be used to even up a non-zero R. We then sort the list of nearest neighbours candidates $(\mathbf{z}_p, \mathbf{x}_q)$ in order of increasing $D(\mathbf{z}_p, \mathbf{x}_q)$ (supposing all colliding pairs have been removed) and in this order we perform the following operations for each nearest neighbour candidate $(\mathbf{z}_p, \mathbf{x}_q)$:

if $c_p < K$ **then**

we assign $\mathbf{z}_p$ to $\mathbf{x}_q$ and increment c_p

else if $c_p = K$ and $R > 0$ **then**

we assign $\mathbf{z}_p$ to $\mathbf{x}_q$, increment c_p and decrement R.

This ensures that the uniformity constraint (5) is satisfied while letting all source patches participate equally during the NNF retrieval.

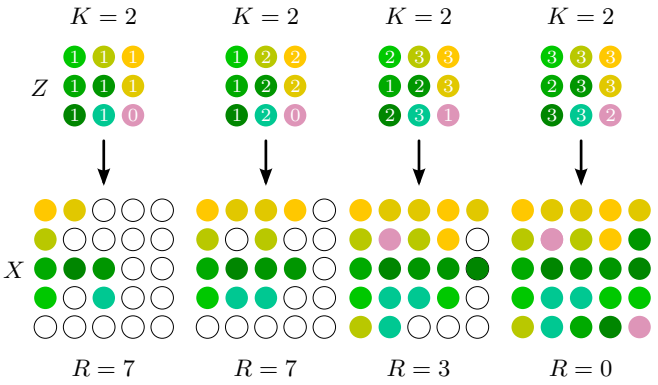


Figure 3: *Reversed construction of nearest-neighbour field. For all patches in the source Z we find best matching candidates in the target X. Since $|Z| < |X|$ and some nearest neighbour candidates can collide with others we need to repeat the retrieval until all patches from X have been assigned their corresponding patches in Z. Patch counters c_p (white numbers), quotient $K = \lfloor |X|/|Z| \rfloor$ and remainder $R = |X| \bmod |Z|$ ensure uniformity of source patch usage (see text for details).*

4.2 Boundary Effects

The algorithm described in the previous section assumes that all patches from the source will be used equally in the target. In our scenario, however, we need to make a distinction between the boundaries (B) and interiors (I) of fluid elements (see Fig. 4). To construct I and B we blur the corresponding alpha masks ($Z_a \rightarrow \mathbb{Z}_\alpha$ and $X_a \rightarrow \mathbb{X}_\alpha$) using Gaussian blur with radius r and apply lower l and upper u opacity thresholds. For Z this yields B_Z: $\mathbb{Z}_\alpha \in (l, u)$ and I_Z: $\mathbb{Z}_\alpha \geq u$ (likewise for X, see Fig. 4). This segmentation lets us restrict the NNF retrieval so that all patches from B_Z are matched to those from B_X and all patches from I_Z to those from I_X. To enforce uniformity (5) in each segment we set $K = |I_X|/|I_Z|$ and $R = |I_X| \bmod |I_Z|$ for all patches in I_Z and $K = |B_X|/|B_Z|$ and $R = |B_X| \bmod |B_Z|$ for all patches in B_Z.

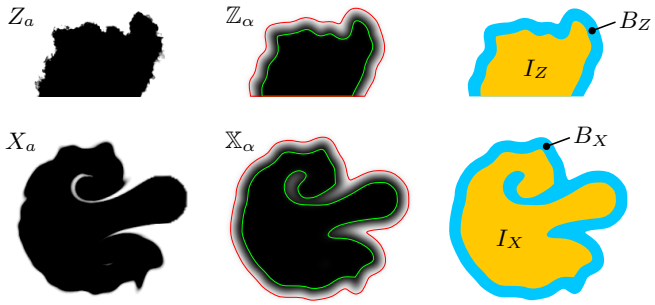


Figure 4: *Construction of boundary B and interior I segments in the source Z and the target X. Input alpha masks Z_a and X_a are blurred ($\mathbb{Z}_\alpha$ and $\mathbb{X}_\alpha$) and then lower (green curve) and upper (red curve) opacity thresholds are applied to obtain B_Z and B_X, and I_Z and I_X.*

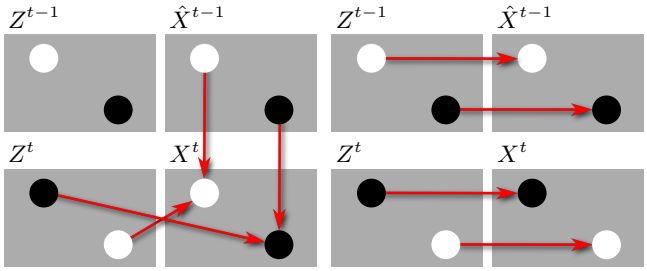
To enable the synthesis of a detailed alpha mask for the target we use Z_a and X_a as additional pixel channels. During each iteration of the modified Kwatra et al. algorithm [2005] a pixel channel corresponding to the target alpha mask X_a is modified along with the

regular color channels X_r, X_g, and X_b. To increase the influence of these additional channels when computing the pixel difference we set their weight to be three times higher than the individual color channels.

For some exemplars a simple boundary/interior distinction might not be sufficient since patches from the outer part of the boundary can still be assigned to the inner part and vice versa. To alleviate this confusion, we add blurred alpha masks $\mathbb{Z}_\alpha$ and $\mathbb{X}_\alpha$ as additional pixel channels. $\mathbb{X}_\alpha$ stays fixed during the synthesis and biases the NNF retrieval so that patches closer to the transition $B_Z \leftrightarrow I_Z$ in the source are more likely to be mapped to the target transition $B_X \leftrightarrow I_X$ and vice versa. We let the user control this bias by adding a special weighting parameter η. Lower η means greater variability in the synthesized boundary effects.

4.3 Temporal Coherence

To ensure temporal coherence (3) Kwatra et al. [2005] iteratively blend the currently synthesized frame with a forward-warped version of the previously synthesized frame $\hat{X}^{t-1}$. Unfortunately, this approach works only for static sources since it enforces similarity only to the previously synthesized frame (see Fig. 5, left). However, we would like to support video exemplars $(Z^t)_{t=1}^M$ as well, so we need a different approach.



[Kwatra et al. 2005] our approach

Figure 5: *Temporal coherence in the case of video exemplar—the original approach of Kwatra et al. [2005] (left) enforces temporal coherence by measuring similarity between the synthesized frame X^t and the forward-warped previously synthesized frame $\hat{X}^{t-1}$. In case of a video exemplar $(Z^t)_{t=1}^M$ the synthesis yields incorrect results since the temporal coherence term enforces continuity in spite of changes between frames Z^t and Z^{t-1}. In our solution (right) we search for patches that are independently similar to the source, both in the forward-warped version of the previous frame and the currently synthesized frame.*

Our approach is similar to what we do for boundary effects: we introduce 4 additional RGBA channels into the source $Z^t(p)$ and target $X^t(q)$ pixels that contain values from collocated pixels in previous frames $Z^{t-1}(p)$ and $\hat{X}^{t-1}(q)$. The additional RGBA channels influence the overall patch similarity (see Fig. 5, right) and thus bias the NNF retrieval to prefer source patches whose appearance is close to both the current and the forward-warped previous frame. For a static exemplar we simply duplicate the content of the regular RGBA channel.

Spatially variable temporal coherence is needed to handle emitters—places where new fluid is spawned. At those places the forward-warping mechanism would incorporate the surrounding empty pixels, but we need to create a new patch of fluid inside the emitter instead. To accomplish this we set λ_q equal to λ for pixels where fluid exists in both the previous and current frames, and set it to zero for pixels where the new fluid appears. These regions can easily be deduced from the current frame mask X_a^t and the forward-warped mask of the previous frame $\hat{X}_a^{t-1}$ (see Fig. 6) using the following equation:

$$\lambda_q = (1 - \max(0, X_a^t(q) - \hat{X}_a^{t-1}(q)))\lambda. \tag{6}$$

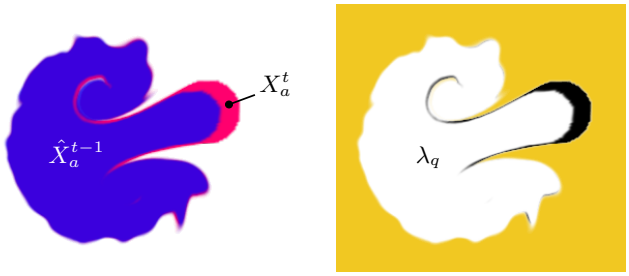


Figure 6: *Allowing new fluid to be injected into the domain by locally down-weighting the temporal coherence term. It becomes a spatially varying function λ_q (right), which is zero where the forward-mapped mask of the previous frame $\hat{X}_a^{t-1}$ does not overlap the current frame's mask X_a^t (left).*

4.4 Joint Formulation

The extensions proposed in Sections 4.1, 4.2 and 4.3 can now be formally combined into one joint optimization problem. We concatenate all additional channels to form a new enriched source $\widetilde{Z} = (Z^t, \mathbb{Z}_\alpha^t, Z^{t-1})$ and new enriched target $\widetilde{X} = (X^t, \mathbb{X}_\alpha^t, \hat{X}^{t-1})$ (see Fig. 7). Now the aim becomes minimizing the following energy:

$$E_s(\widetilde{Z}, \widetilde{X}, \lambda, \eta) = \sum_{q \in \widetilde{X}} \min_{p \in \widetilde{Z}} D(\tilde{\mathbf{z}}_p, \tilde{\mathbf{x}}_q, \lambda_q, \eta) \tag{7}$$

subject to uniformity constraint (5). Here λ_q is the spatially variant weight for temporal coherence, η is the weight for boundary coherence, and $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{z}}$ denote patches with 9 channels per pixel:

$$\begin{aligned} \tilde{\mathbf{x}} &= (\mathbf{x}_{rgb}^t, \mathbf{x}_a^t, \mathbf{x}_\alpha^t, \hat{\mathbf{x}}_{rgb}^{t-1}, \hat{\mathbf{x}}_a^{t-1}) \\ \tilde{\mathbf{z}} &= (\mathbf{z}_{rgb}^t, \mathbf{z}_a^t, \mathbf{z}_\alpha^t, \mathbf{z}_{rgb}^{t-1}, \mathbf{z}_a^{t-1}) \end{aligned} \tag{8}$$

where $\mathbf{x}_{rgb}^t$ denotes the color and $\mathbf{x}_a^t$ the alpha mask of the currently synthesized frame, $\mathbf{x}_\alpha^t$ is the blurred alpha mask of the current frame, $\hat{\mathbf{x}}_{rgb}^{t-1}$ is the color and $\hat{\mathbf{x}}_a^{t-1}$ the alpha mask of the previous frame (likewise for $\tilde{\mathbf{z}}$). Finally D is the distance measure between patches $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{z}}$ (see Fig. 7):

$$\begin{aligned} D(\tilde{\mathbf{z}}, \tilde{\mathbf{x}}, \lambda, \eta) = &||\mathbf{z}_{rgb}^t - \mathbf{x}_{rgb}^t||^2 \\ &+ 3||\mathbf{z}_a^t - \mathbf{x}_a^t||^2 \\ &+ \eta||\mathbf{z}_\alpha^t - \mathbf{x}_\alpha^t||^2 \\ &+ \lambda||\mathbf{z}_{rgb}^{t-1} - \hat{\mathbf{x}}_{rgb}^{t-1}||^2 \\ &+ 3\lambda||\mathbf{z}_a^{t-1} - \hat{\mathbf{x}}_a^{t-1}||^2 \end{aligned} \tag{9}$$

To minimize (7) we use the EM-like multi-scale algorithm described by Kwatra et al. [2005] and Wexler et al. [2007]. The only necessary modifications are that the source and target patches contain pixels with 9 weighted channels (4 are synthesized and 5 are for guidance, c.f. Fig. 7) and that the NNF retrieval phase is replaced by our method supporting uniform patch usage (see Sections 4.1 & 4.2). Temporal coherence is implicitly incorporated thanks to additional pixel channels—there is no need to use a special algorithm for controllable texture synthesis as described by Kwatra et

al. The first frame is synthesized with $\lambda = 0$ and then we proceed in frame-by-frame order analogous to Kwatra et al. An interesting side-effect of the uniformity constraint is that the algorithm does not require any specific initialization. It is possible to use either the initialization described in Kwatra et al. or any other (e.g., zeroing). The algorithm always comes up with a solution that uses all source patches an equal number of times.

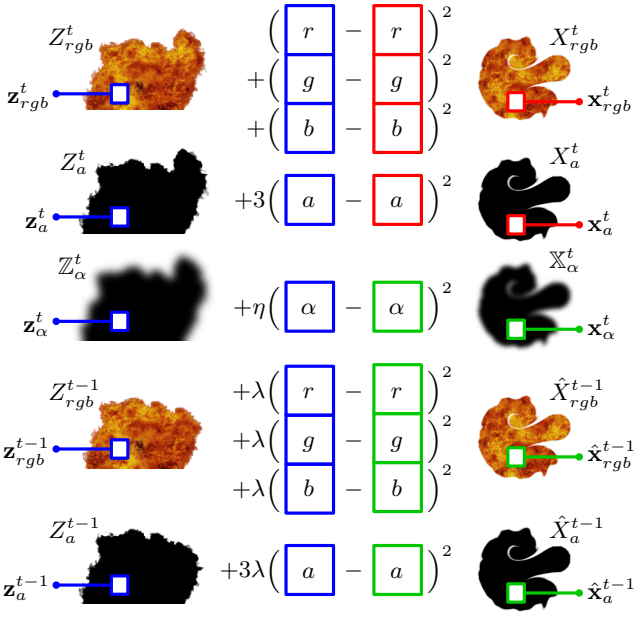


Figure 7: *Measuring the distance between a source and target pixel within patches $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{x}}$. The distance is the sum of 9 terms corresponding to the following pixel channels: 4 for the current frame ($[Z, X]^t_{rgba}$), 4 for the previous frame ($[Z, \hat{X}]^{t-1}_{rgba}$), and 1 for the blurred alpha mask of the current frame $[\mathbb{Z}, \mathbb{X}]^t_\alpha$. Channels X^t_{rgba} (red squares) are modified during the synthesis while the others (green squares) remain fixed. Parameter λ is the spatially variant weight that enforces temporal coherence and η is the boundary coherence weight.*

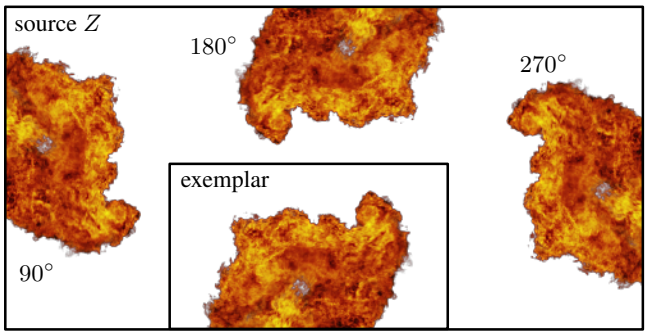


Figure 8: *Rotating the input exemplar to ensure that boundaries of all orientations are available in the source.*

4.5 Further Extensions

The method can be extended to handle arbitrarily rotated patches by having the NNF retrieval phase look over the space of rotations in addition to translations. The patch counting mechanism and NNF construction would remain unchanged. Although such an extension could improve the appearance on some specific flows (e.g., pure rotation in Fig. 2) it can significantly increase the overall processing time. In practice we use a simpler approximation with notably lower computational overhead, originally proposed in [Lukáč et al. 2013]. We pre-rotate the exemplar by 90°, 180°, and 270° and perform synthesis with this enriched source (see Fig. 8).

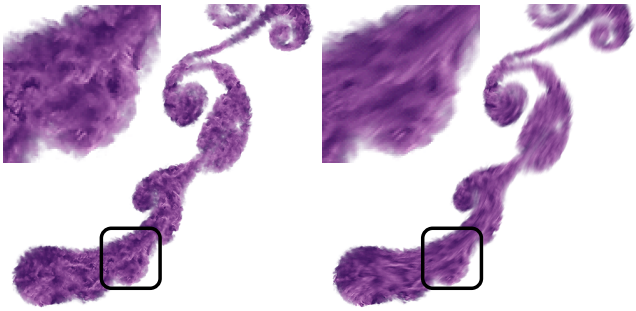


Figure 9: *Adding synthetic motion blur to improve the fidelity of the resulting fluid animation—the direction and magnitude of motion vectors in the target motion field F are used to perform line convection with an anisotropic Gaussian filter.*

The fidelity of the resulting animation can further be improved using synthetic motion blur. This additional effect can easily be implemented since we know the exact motion field of the target animation; thus we can perform line convection using anisotropic Gaussian filter in a direction and length given by the target motion field (see Fig. 9).

5 Results

We have implemented our technique in C++. To achieve feasible performance we accelerated the NNF retrieval phase described in Section 4.1 using PatchMatch with integrated support for masking [Barnes et al. 2009]. To further accelerate this performance bottleneck we used a parallel tiled algorithm [Barnes et al. 2010]. With this optimization it usually takes around 5 minutes to synthesize one 1Mpix frame on a 3GHz CPU with 4 cores.

To prepare data sets we implemented a custom fluid simulator [Stam 1999] that allows an artist to quickly design desired fluid animations in real-time. Using this tool we created 7 different fluid animations with different motion properties to be used as targets, and selected 10 different fluid elements (7 static images and 3 videos) with variable appearance and complexity to be used as exemplars. To produce the results we used the following parameter setting: patch size 5×5, temporal coherence $\lambda = 0.2$, richness of boundary effects $\eta = 3$, radius for Gaussian blur: $r = 3$ (this setting might change according to the resolution of the target animation), and thresholds: $l = 0.1$, $u = 0.9$ to extract boundary region from of the blurred target alpha channel.

First we synthesized a single fluid animation using multiple exemplars (see Fig. 10). Our approach produced convincing results despite notable visual differences between individual sources. For the fire exemplars in Figures 1 and 11 we compared the static source to the video source. From the comparison (see supplementary video) it is visible that our approach produces convincing results for both cases. The advantage of the video source is that it nicely enhances the motion richness of the target animation.

We also compared our approach with the original Kwatra et al. [2005] method as well as Bénard et al. [2013] and Browning

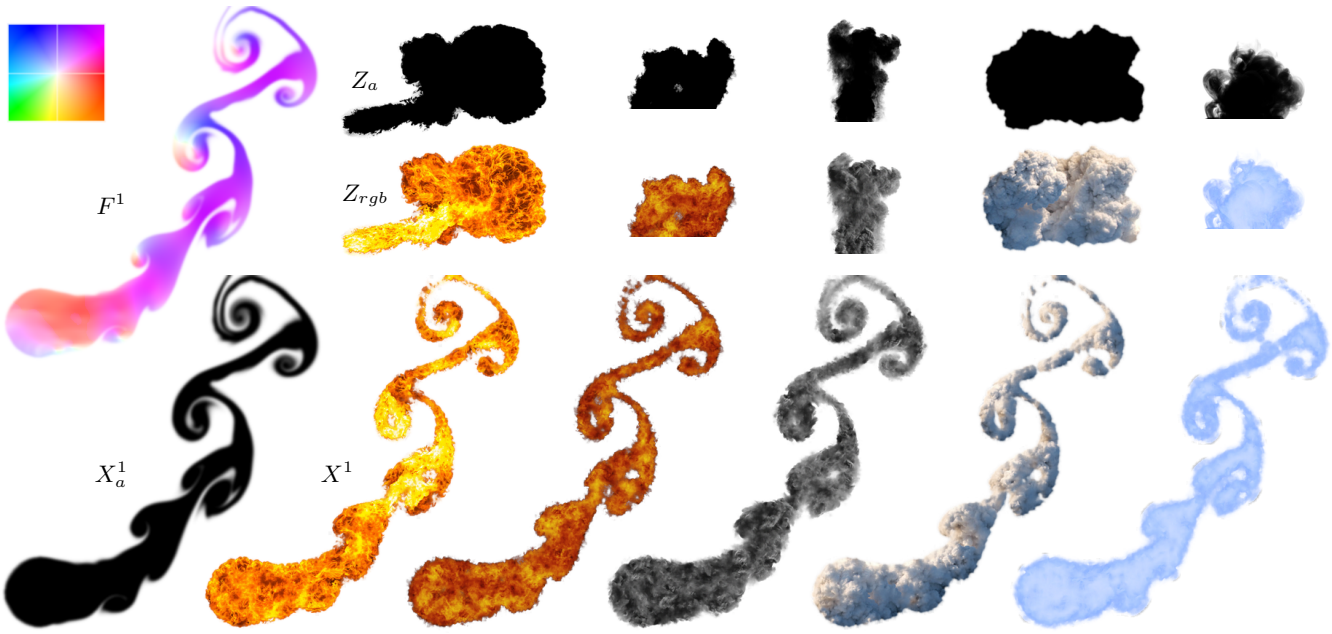


Figure 10: *Results and comparison: various fluid exemplars consisting of color Z_{rgb} and alpha mask Z_a (both static images) were used to synthesize a target fluid animation X specified by a sequence of flow fields $(F^t)_{t=1}^{180}$ and alpha masks $(X_a^t)_{t=1}^{180}$ (showing only a single frame, see supplementary video for the whole sequence). Note how our approach produces convincing results despite varying source complexity. Fire blast exemplar (2nd column) © Corteck via flickr, volcano smoke exemplar (5th column) © Arnar Thorisson.*

et al. [2014], two current state-of-the-art techniques in flow-guided appearance transfer. For each method we used the same fluid animation as well as fluid exemplars as in Fig. 10 to enable side-by-side comparison. Since Kwatra et al. does not natively support synthesis with boundary effects we decided to illustrate its behavior by using our own approach, but with the original NNF retrieval phase—not our improved method that enforces uniform patch usage. Setup for Bénard et al. was very close to our method. For style input S we used the blurred alpha channel of our exemplar Z_a, for style output $\hat{S}$ we used the RGB channels of our exemplar Z_{rgb}, and finally for input image I we used our blurred target alpha mask X_a. For Browning et al. we had to prepare a set of keyframes (every 10th frame) that were synthesized using our approach as independent frames (i.e., we set $\lambda = 0$).

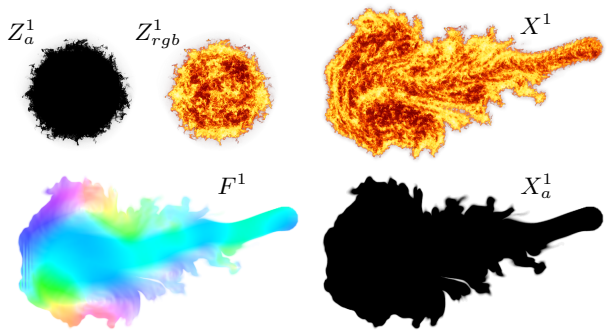


Figure 11: *Synthesis with a video exemplar: input flow fields F with alpha masks X_a and fluid exemplar Z_a and Z_{rgb} was used to synthesize fluid animation X. The fireball sequence $(F^t)_{t=1}^{190}$ was used to compare a static source and a video source $(Z^t)_{t=1}^{120}$ (see supplementary video for the comparison).*

We ran these methods on all 180 frames of the test sequence in Fig. 10. Static frames from this sequence are presented in Fig. 13 and complete videos are in supplementary materials. It is clear from the results that the original NNF retrieval used by Kwatra et al. produces severe gradual wash-out and thus quickly diverts the appearance of the resulting sequence from the appearance of the exemplar. In the results of Bénard et al. the gradual wash-out is also visible, however, it is not as apparent thanks to the offset histogram matching used in the original method. However, several disturbing temporal artifacts are visible such as high-frequency popping, and the boundary effects are not as detailed as in our results. The results of Browning et al. exhibit the closest appearance to the exemplar. There is no gradual wash-out visible since the method relies on frequent keyframes that were synthesized using our method. However, the results show ghosting and a disturbing pulsation effect that is more visible in the video.

Finally, to further demonstrate practical utility of our approach we produced 4 compositions where different fluid animations were synthesized using realistic fluid exemplars and then combined with an environment to create a desired visual effect (see Figures 1 and 12 and supplementary video). Such practical results were impossible to achieve using previous techniques as they either suffer from gradual wash-out or produce disturbing temporal artifacts.

6 Limitations and Future Work

We found that in practice our method performs very well. It is quite robust, being agnostic to parameter settings, type of input fluid simulation and the choice of exemplar. Nevertheless, there are several limitations that we would like to mention.

In cases where the source exemplar is substantially bigger than the target, and when it contains large areas with noticeably smoother or lower contrast patches, our technique cannot always fully avoid gradual wash-out since the harmful preference for smoother

Figure 12: *A more complex composition created by an artist using our system (top). Three different flow fields F with alpha masks X_a and exemplars Z_{rgb} with alpha channels Z_a (bottom) were used to synthesize the resulting fluid animations. Firepit painting © Jakub Javora.*

patches [Newson et al. 2014] can still prevail. This is a challenging situation that has great potential for further investigation.

Our technique does not take into account additional flow-field parameters such as curvature, divergence or Jacobian. Those can be computed by the simulation and can be used to guide the appearance of synthesized texture locally (like in [Narain et al. 2007]) to better convey realistic physical properties of the given fluid element. Similar limitations hold for shading, self shadowing, or volumetric effects. Those are baked into the exemplar for some particular lighting conditions, but they might not be realistic in a different environment. Both issues are good candidates for follow-up work.

Since our algorithm synthesizes the i-th target frame using the i-th frame of the video exemplar we need the video exemplar to have at least as many frames as the output sequence. If it is shorter, some additional looping is necessary to avoid hard jumps in appearance. Techniques that produce looped sequences from unlooped footage [Schödl et al. 2000; Liao et al. 2013] can help the user prepare input data in this case. A related issue is motion in the input sequence. When its direction or speed do not match the target animation our method can produce unsatisfactory results. In this case some additional stabilization of the source would be necessary.

7 Conclusion

We have presented a novel approach to appearance transfer for fluid animations. Our technique is the first that performs flow-guided texture synthesis that convincingly preserve the appearance of realistic fluid exemplars, while at the same time avoiding disturbing temporal artifacts. We have demonstrated the practical utility of our technique in realistic scenarios, giving practitioners a new option for creating fluid-based special effects. We also believe that our novel constrained formulation, which ensures uniform patch use, should inspire future research that will improve the quality of patch-based texture synthesis and related image editing techniques.

Acknowledgements

We would like to thank Jakub Javora for creating the compositions in Figures 1 and 12. We are grateful to Pierre Bénard, Michael Bußler, Mark Browning, Adam Finkelstein, Alexandre Kaspar, and Johannes Kopf for helping us with evaluation. Thanks also go to all anonymous reviewers for their constructive comments. This research was funded by Adobe and has been supported by the Technology Agency of the Czech Republic under research program TE01020415 (V3C), by the Czech Science Foundation under research program P202/12/2413 (OPALIS), and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS13/214/OHK3/3T/13 (Research of Progressive Computer Graphics Methods). Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

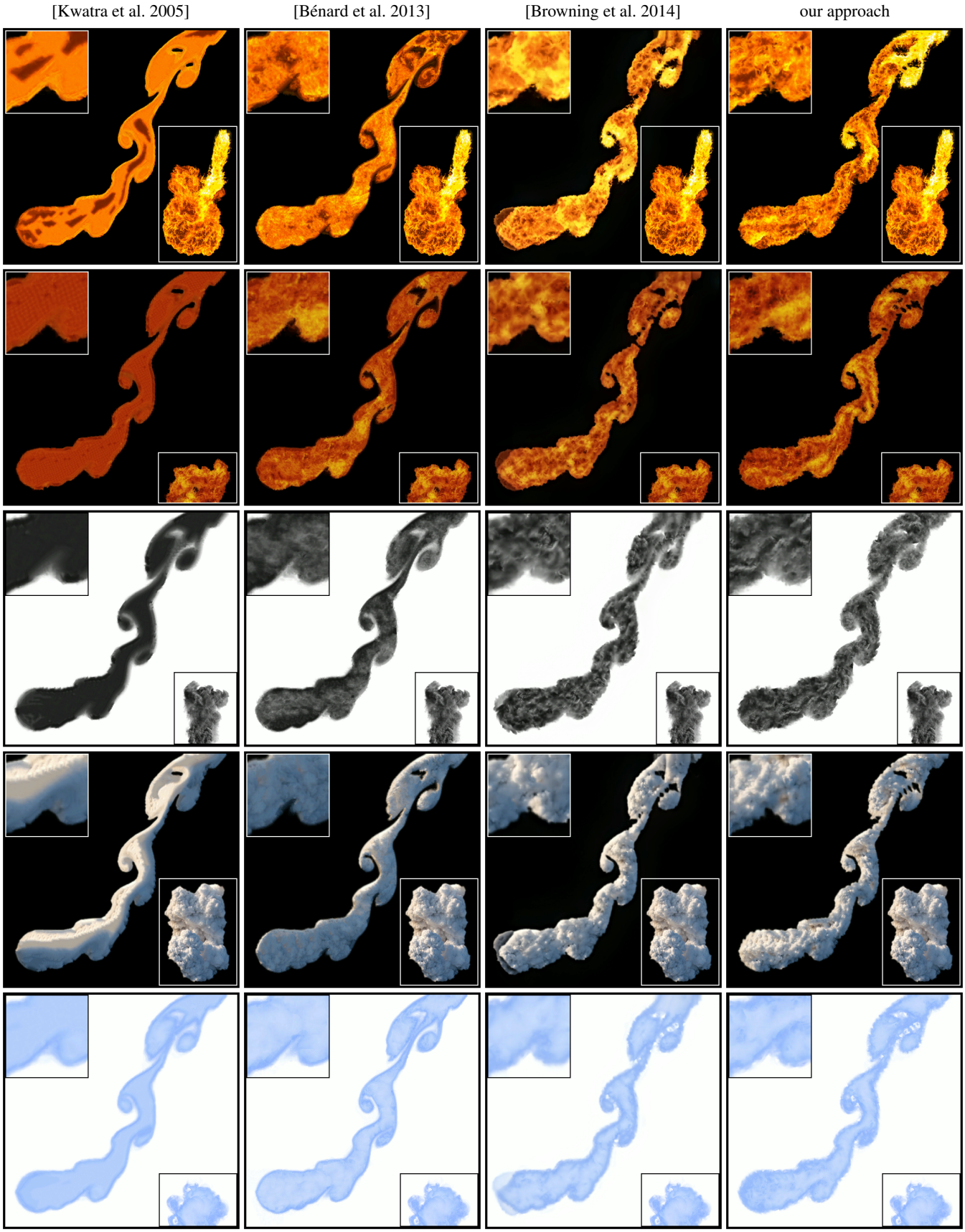


Figure 13: *Results and comparison (see text for details): a single frame (No. 135) extracted from a longer sequence synthesized using [Kwatra et al. 2005], [Bénard et al. 2013], [Browning et al. 2014], and our approach on five different exemplars of fluid elements (shown in the bottom right corner in each view). The top left corner shows a detail of the synthesized result. Fire blast exemplar (1st row)* © *Corteck via flickr, volcano smoke exemplar (4th row)* © *Arnar Thorisson.*

References

BARGTEIL, A. W., SIN, F., MICHAELS, J. E., GOKTEKIN, T., AND O'BRIEN, J. F. 2006. A texture synthesis method for liquid animations. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 345–351.

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics 28*, 3, 24.

BARNES, C., SHECHTMAN, E., GOLDMAN, D. B., AND FINKELSTEIN, A. 2010. The generalized PatchMatch correspondence algorithm. In *Proceedings of European Conference on Computer Vision*, 29–43.

BÉNARD, P., COLE, F., KASS, M., MORDATCH, I., HEGARTY, J., SENN, M. S., FLEISCHER, K., PESARE, D., AND BREEDEN, K. 2013. Stylizing animation by example. *ACM Transactions on Graphics 32*, 4, 119.

BHAT, K. S., SEITZ, S. M., HODGINS, J. K., AND KHOSLA, P. K. 2004. Flow-based video synthesis and editing. *ACM Transactions on Graphics 23*, 3, 360–363.

BOUSSEAU, A., NEYRET, F., THOLLOT, J., AND SALESIN, D. 2007. Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics 26*, 3, 104.

BROWNING, M., BARNES, C., RITTER, S., AND FINKELSTEIN, A. 2014. Stylized keyframe animation of fluid simulations. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering*, 63–70.

CHEN, J., AND WANG, B. 2010. High quality solid texture synthesis using position and index histogram matching. *The Visual Computer 26*, 4, 253–262.

DARABI, S., SHECHTMAN, E., BARNES, C., GOLDMAN, D. B., AND SEN, P. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics 31*, 4, 82.

HAN, J., ZHOU, K., WEI, L.-Y., GONG, M., BAO, H., ZHANG, X., AND GUO, B. 2006. Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer 22*, 9–11, 918–925.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *Proceedings of SIGGRAPH 2001*, 327–340.

KASPAR, A., NEUBERT, B., LISCHINSKI, D., PAULY, M., AND KOPF, J. 2015. Self tuning texture optimization. *Computer Graphics Forum 34*, 2, 349–360.

KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics 26*, 3, 2.

KWATRA, V., SCHÖDL, A., ESSA, I. A., TURK, G., AND BOBICK, A. F. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3, 277–286.

KWATRA, V., ESSA, I. A., BOBICK, A. F., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics 24*, 3, 795–802.

KWATRA, V., ADALSTEINSSON, D., KIM, T., KWATRA, N., CARLSON, M., AND LIN, M. C. 2007. Texturing fluids. *IEEE Transactions on Visualization and Computer Graphics 13*, 5, 939–952.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics 25*, 3, 541–548.

LIAO, Z., JOSHI, N., AND HOPPE, H. 2013. Automated video looping with progressive dynamism. *ACM Transactions on Graphics 32*, 4, 77.

LUKÁČ, M., FIŠER, J., BAZIN, J.-C., JAMRIŠKA, O., SORKINE-HORNUNG, A., AND SÝKORA, D. 2013. Painting by feature: Texture boundaries for example-based image creation. *ACM Transaction on Graphics 32*, 4, 116.

MAX, N., CRAWFIS, R., AND WILLIAMS, D. 1992. Visualizing wind velocities by advecting cloud textures. In *Proceedings of IEEE Conference on Visualization*, 179–183.

NARAIN, R., KWATRA, V., LEE, H.-P., KIM, T., CARLSON, M., AND LIN, M. C. 2007. Feature-guided dynamic texture synthesis on continuous flows. In *Proceedings of Eurographics Symposium on Rendering*, 361–370.

NEWSON, A., ALMANSA, A., FRADET, M., GOUSSEAU, Y., AND PÉREZ, P. 2014. Video inpainting of complex scenes. *SIAM Journal of Imaging Science 7*, 4, 1993–2019.

NEYRET, F. 2003. Advected textures. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 147–153.

PERLIN, K. 1985. An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3, 287–296.

REEVES, W. T. 1983. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2*, 2, 91–108.

ROSENBERGER, A., COHEN-OR, D., AND LISCHINSKI, D. 2009. Layered shape synthesis: Automatic generation of control maps for non-stationary textures. *ACM Transactions on Graphics 28*, 5, 107.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *SIGGRAPH Conference Proceedings*, 489–498.

SHECHTMAN, E., RAV-ACHA, A., IRANI, M., AND SEITZ, S. M. 2010. Regenerative morphing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 615–622.

SIMAKOV, D., CASPI, Y., SHECHTMAN, E., AND IRANI, M. 2008. Summarizing visual data using bidirectional similarity. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

STAM, J. 1999. Stable fluids. In *SIGGRAPH Conference Proceedings*, 121–128.

WEI, L.-Y., HAN, J., ZHOU, K., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Inverse texture synthesis. *ACM Transactions on Graphics 27*, 3.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 3, 463–476.

YU, Q., NEYRET, F., BRUNETON, E., AND HOLZSCHUCH, N. 2011. Lagrangian texture advection: Preserving both spectrum and velocity field. *IEEE Transactions on Visualization and Computer Graphics 17*, 11, 1612–1623.