Mid-level Smoke Control for 2D Animation

Alfred Barnat
Carnegie Mellon University

Zeyang Li
Stanford University*

James McCann
Adobe Systems, Inc.*

Nancy S. Pollard
Carnegie Mellon University

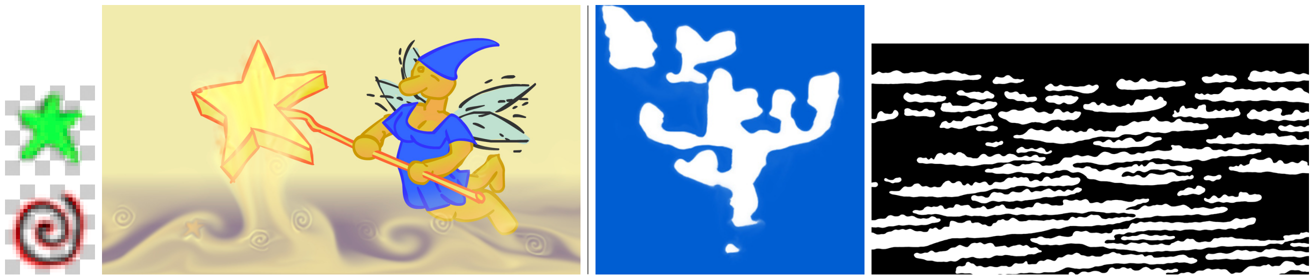


Figure 1: Our mid-level fluid control system allows artists to specify local smoke behavior by providing density motifs (**left**). The resulting fluid can be integrated with 2D animations (**center left**). In this case, we have recolored the smoke to match the wand colors. Other effects our system can generate (**center right**) include global textures such as clouds reminiscent of those found in Katsushika Hokusai's prints (**right**).

Abstract

In this paper we introduce the notion that artists should be able to control fluid simulations by providing examples of expected local fluid behavior (for instance, an artist might specify that magical smoke often forms star shapes). As our idea fits between high-level, global pose control and low-level parameter adjustment, we deem it mid-level control. We make our notion concrete by demonstrating two mid-level controllers providing stylized smoke effects for two-dimensional animations. With these two controllers, we allow the artist to specify both density patterns, or *particle motifs*, which should emerge frequently within the fluid and global *texture motifs* to which the fluid should conform. Each controller is responsible for constructing a stylized version of the current fluid state, which we feed-back into a global pose control method. This feedback mechanism allows the smoke to retain fluid-like behavior, while also attaining a stylized appearance suitable to integration with 2D animations. We integrate these mid-level controls with an interactive animation system, in which the user can control and keyframe all animation parameters using an interactive timeline view.

Index Terms: I.3.4 [Computer Graphics]: Graphics Utilities—Graphics packages; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1 Introduction

Current approaches for controlling fluid simulation fall into two bins: low-level and high-level. Low-level controls, e.g. Foster and Metaxas [6], give artists tools to tweak basic simulation parameters or create local effects. These methods are useful when an artist is trying to model a specific fluid (honey instead of water, for instance), or get a general behavior (explosion, splash). In contrast, high-level controls, e.g. Treuille et al. [18], modify fluid behavior to attain target poses. These techniques give artists and directors explicit global control over fluid pose, perhaps at the expense of realism.

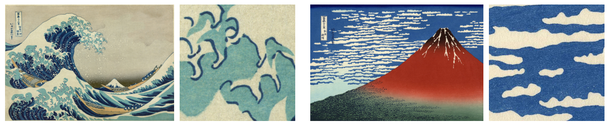


Figure 2: Katsushika Hokusai's iconic wood-block prints feature fluids which have a striking and consistent local appearance that would be hard to duplicate with current high-level or low-level fluid control methods.

But what of an artist who has a specific notion of local fluid behavior, but simply wishes for the global fluid pose to evolve naturally? For instance, the water and clouds appearing in classic wood-block prints by Hokusai (Figure 2) have a striking appearance that goes beyond simple stylization and would be hard to duplicate in a classic high- or low-level control framework. For this, a new type of fluid control is needed: *mid-level control*.

We consider a mid-level control approach to be one that maintains the *ambiguity* of an artist's intent – controlling specific local fluid behaviors while allowing the global fluid pose to evolve naturally due to physics. Thus, any mid-level control approach must make two choices: first, what sort of local properties artists should be able to specify; and, second, how these properties will be maintained.

There are of course many valid answers to these questions, but – in this paper – we will describe a specific mid-level smoke control framework designed for integration with two-dimensional animations (Figure 1). We choose to let artists specify fluid behavior by providing local density patterns, or *particle motifs*, that should appear frequently over the course of the simulation (for instance, skulls in a dangerous miasma, or spirals and stars in a powerful magic aura), as well as a globally applied but locally enforced *texture motif*, which controls the overall appearance of the fluid (as in the appearance of Hokusai's clouds). We cause these density patterns to emerge by using a continuously-updated stylized version of the present density field as a target for a global smoke-control method (Figure 3). This basic premise – using a stylized version of the current fluid state as the target of a global control method – both works well in our specific system and seems like a powerful blueprint for implementing future mid-level controllers.

We demonstrate the applicability of our approach by integrating

*Work performed while at Carnegie Mellon University.

it with an interactive animation system. The animation is controlled by setting keyframe values for control parameters, and the system automatically renders the animation as the user is working. We demonstrate the effectiveness of this system by adding controlled smoke to several 2D animations.

2 Background

Fluid animation in graphics has a rich history. Below, we highlight a few works which are the most relevant to our mid-level control scheme. For general information on simulated fluids in graphics, we refer the reader to a recent survey paper [17].

Our smoke simulations use the global control method of Fattal and Lischinski [5] as a primitive. Where Fattal and Lischinski demonstrated this approach with user-specified keyframes, we automatically generate and continuously update the fluid target.

Several alternative global control methods exist, though none are as well suited to our needs as the above. One approach, given by Treuille et al. [18] and later refined using the adjoint method [10], works by finding optimal driving forces that will allow smoke (or liquid) to match a sequence of user-specified keyframes. As these methods require knowledge of targets several frames in the future, they are not well suited to our needs. A different approach, presented by Lin and Yizhou [15], defines an implicit surface for both the target shape and current density fields, and adds velocities in order to push the density surface towards the target. This method is not especially well suited for use with our target density fields, since they do not necessarily define discrete objects and instead define a field of density values to be matched.

The control method of Schpok et al. allows users to manipulate flows using a vocabulary of mid-level features extracted from the simulation [13]. Similar control may be achieved by using a simplified basis for the simulated velocity field, as in the method of Angelidis et al. [1]. Here, the velocity field is represented as a set of vortex filaments, allowing an animator to more easily understand the effect of any modifications. However, in both of these cases, users are limited to local structures they do not select.

Another consideration in artistic fluid control is the ability to preserve overall behavior between low- and high-resolution simulations. Nielsen and Christensen offer a solution to this by using a low-resolution simulation to guide the global behavior of a corresponding high-resolution simulation, while allowing high-frequency details to emerge [12].

Stylized fluid rendering approaches seek to present the output of a fluid animation in a given style. One such approach is the use of billboard particles for cartoon smoke rendering [9, 14]. Another approach is to use a texture synthesis technique controlled by the velocity field produced by the simulation [2, 7, 8, 11]. Perhaps the most similar of these techniques to our own is that of Ma et al. [8], which uses texture synthesis to sequentially apply a velocity exemplar, or motif, to each frame of an animation. This technique is similar in that the textured velocity field for each frame is seeded by advecting the textured field from the previous frame. However, texturing the velocity field instead of the density field limits the artist's ability to specify desired density configurations, and more importantly, the texturing is unable to affect any underlying simulation. In all of these approaches, there is no feedback loop – in our system, the stylization is more than just cosmetic, it actually changes long-term fluid behavior.

3 Contributions

We introduce a technique for artist-directed fluid stylization with feedback into the underlying simulation. We use the current fluid state in order to determine how and where motifs appear in the fluid, with the goal being to move the fluid incrementally to match the artist's desired look (Figure 3). The changes caused in the fluid by these motifs in turn feed back into the selection mechanism used to place motifs in future frames. We have developed two mid-level fluid control techniques making use of this feedback mechanism.

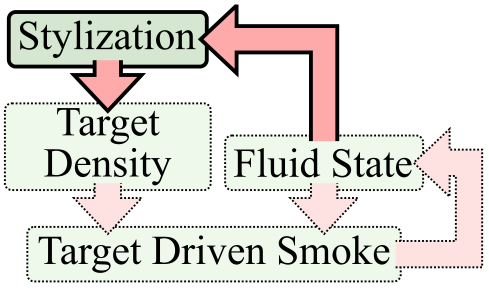


Figure 3: We achieve mid-level control by adding stylization feedback (solid border) to an existing smoke control method [5] (dotted border).

The first of these techniques places motif particles, representing shapes the artist wishes to appear in the fluid. Both the initial placement of these particles and their motion is determined by the fluid state. The presence of these particles then affect the local behavior of the fluid, feeding back into the underlying simulation.

Our second technique works across the entire fluid, attempting move the fluid to match an artist-specified texture motif. This is done by using a texture transfer technique to construct a stylized version of the fluid state. This stylized fluid is then used to guide the underlying simulation, thus completing the feedback loop.

4 Method

We build our system on the global smoke control method of Fattal and Lischinski [5]. Instead of using predetermined keyframes, we feed-back a stylized version of the density field as the smoke target (Figure 3). The stylized density field can be generated using either a collection of motif particles or a motif texture. Motif particles cause localized regions of the fluid which already correspond roughly to one of a set of user-defined patterns to match the pattern more closely. Motif textures work globally, causing the fluid to conform to the patterns present in a user-defined sample image.

4.1 Smoke Control

Since our system uses target-driven smoke [5] as a primitive, we briefly summarize its operation. It is the goal of target-driven smoke to drive the current density field ρ toward some target density field ρ^*. To do so, driving forces $\mathbf{F}(\rho,\rho^*)$ and damping forces $v_d\mathbf{u}$ are introduced into the equations of fluid motion:

$$\mathbf{u}_t = -\mathbf{u}\cdot\nabla\mathbf{u} - \nabla p + v_f\mathbf{F}(\rho,\rho^*) - v_d\mathbf{u} + \mathbf{f} \quad (1)$$

$$\nabla\cdot\mathbf{u} = 0 \quad (2)$$

Here, $\mathbf{u}$ is fluid velocity, p is pressure, $\mathbf{f}$ is external force, and v_f and v_d are constants giving strength of control and damping.

The driving force $\mathbf{F}(\rho,\rho^*)$ moves fluid along the normalized gradient of the blurred target density $\tilde{\rho}^*$:

$$\mathbf{F}(\rho,\rho^*) \equiv \tilde{\rho}\frac{\nabla\tilde{\rho}^*}{\tilde{\rho}^*} \quad (3)$$

(The $\tilde{\rho}$ factor serves to reduce forces where no density exists.)

Additionally, a *gathering term*, $\mathbf{G}(\rho,\rho^*)$ is introduced to the density advection equation to combat numerical dissipation:

$$\rho_t = -\mathbf{u}\cdot\nabla\rho + v_g\mathbf{G}(\rho,\rho^*) \quad (4)$$

Where $\mathbf{G}(\rho,\rho^*)$ is defined to reduce error in matching the target density (that is, $\rho-\rho^*$), wherever there exists density in the vicinity of the target (thus the $\rho\tilde{\rho}^*$ factor):

$$\mathbf{G}(\rho,\rho^*) \equiv \nabla\cdot[\rho\tilde{\rho}^*\nabla(\rho-\rho^*)] \quad (5)$$

4.2 Motif Particles

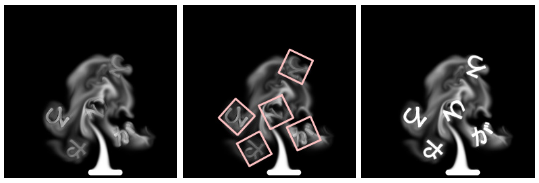


Figure 4: **Left**, a frame from an animation using our motif particle method. **Middle**, motif particle locations in boxes. **Right**, the stylized density field (used as the control target for the next frame).

Our motif particle method produces a target density field that will cause artist-specified patterns to emerge naturally in the fluid. The artist specifies these patterns, or motifs, as a set of small images of desired fluid density. It is the job of our motif particle method to select locations and orientations where these motifs will emerge. To do so, our method uses motif particles to match and track local smoke features that resemble these motifs (Figure 4). These motif particles are blended with the current density field to produce the control target. This has the effect of causing motifs to form in the smoke and persist until destroyed by turbulence.

4.2.1 Motif Matching

It is important for motif particles to appear only in regions already somewhat similar in appearance to their motifs. We find such positions in two steps. First, we find possible good matches by using a simplified rotation-invariant matching scheme inspired by ring projection [4]: we compute a descriptor at each point of the fluid grid by computing the average density at four radii (Figure 6). This descriptor is compared to a similar descriptor computed over each motif, and locations which are above a distance threshold are discarded. Then, for those locations that pass the rotation-invariant matching, we check 32 rotations using an opacity-weighted euclidian distance operator over all pixels in the motif to see if there are good matches to the fluid among them. If any match we find is good enough (i.e. under a user-defined threshold), we introduce a motif particle to track the location and rotation of the match. In order to avoid creating overlapping particles, we do not consider regions already covered by motif particles during this matching process.

In practice the initial matching threshold must be carefully tuned according to the motif complexity. If the motif is a simple shape which is likely to appear occasionally without help, then the threshold can be low. For more complex shapes, a higher threshold is needed. It is also possible to mask out parts of a motif image using the alpha channel, so that they do not contribute to the motif matching. For instance, if a motif is uniformly colored with the shape created only by the alpha channel, it will match any area of the fluid with the correct color regardless of how well the fluid conforms to its shape.

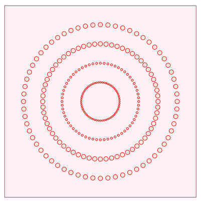


Figure 6: In our first motif-matching phase we use a rotation-invariant descriptor consisting of the average of 32 samples at each of four radii, as pictured. Sample locations are represented by small circles and the region being matched (e.g. a particle motif) is represented by the square background.

4.2.2 Motif Particle Blending

We initialize the control target, ρ^*, with the current density field, then blend the motif particles, M, one at a time. The blending function is user-specified combination of `over` blending (when $\beta = 1, \gamma = 0$) and `add` blending (when $\beta = 0, \gamma = 1$). The result is clamped to the valid range of density values, $[0,1]$:

$$\rho^*(x) \leftarrow (1 - \beta M_\alpha(x))\rho^*(x) + (\gamma + \beta) M_\alpha(x) M_\rho(x) \tag{6}$$

Here, the motif represented by motif particle M has density $M_\rho(x)$ and opacity $M_\alpha(x)$ at location x when translated and rotated to M's position.

It is important to note that particle motifs are never blended directly into the density field, either during simulation or before outputting the density field. Instead, they are blended into a copy of the density field which is used as a target for target-driven smoke in the current frame and then discarded.

4.2.3 Motif Particle Advection

When the density field is advected, we also update our motif particles based on the fluid motion. To do so, we solve for the linear and angular velocities that best match the motion of the smoke, then update the motif particle position based on these velocities. That is, given a motif particle centered at c whose opacity at point x is given by $M_\alpha(x)$ we compute linear velocity v and angular velocity ω which are close, in opacity-weighted squared difference, to the fluid velocity $\mathbf{u}$:

$$\operatorname*{argmin}_{\omega, v} \sum_x M_\alpha(x) \left(\omega \operatorname{perp}(x - c) + v - \mathbf{u}(x)\right)^2 \tag{7}$$

Here, perp(x) is x rotated by 90 degrees.

We found that using the least squares approximation for velocity given in Equation 7 produced more visually pleasing results than other approaches we tried, such as simply averaging velocities of a small number of points, especially in the case of asymmetric particles like the Japanese Hiragana in Figure 4. Our least squares method of evaluation also avoids inaccuracies where particles brush against areas of high velocity.

4.2.4 Motif Particle Revalidation

Over time, motif particles may no longer match the underlying fluid (if, for instance, turbulence disrupts the motif). We recheck the distance function for every motif particle at the beginning of each time step. We use the same rotation-invariant descriptor that was used for the initial match, so that the distances are consistent between the initial match and revalidation. If the distance is found to be above some user-defined threshold, the particle is marked for removal. This threshold is independent from the initial matching threshold, and may be set higher in order to allow the fluid to be somewhat disrupted without causing the particle to be removed.

In addition, we must remove any overlapping particles in order to prevent clumps of particles from forming in convergent areas of the fluid. To do this, we track particle age, and check for any particles which are overlapping older particles. This can be accomplished during motif blending by tracking, for each pixel, whether a particle has been drawn covering the pixel. If we find that any particle overlaps one which has already been drawn, we mark it for removal. Particles marked for removal are checked last, so that they will never cause another particle to be marked for removal.

We do not remove particles instantly, as this creates a noticeable "popping" effect. Instead, we fade the influence of the particle linearly to zero over a user-specified duration d:

$$M_\alpha(x) \leftarrow \left(1 - \frac{t - t_0}{d}\right) M_\alpha^{\text{orig}}(x) \tag{8}$$

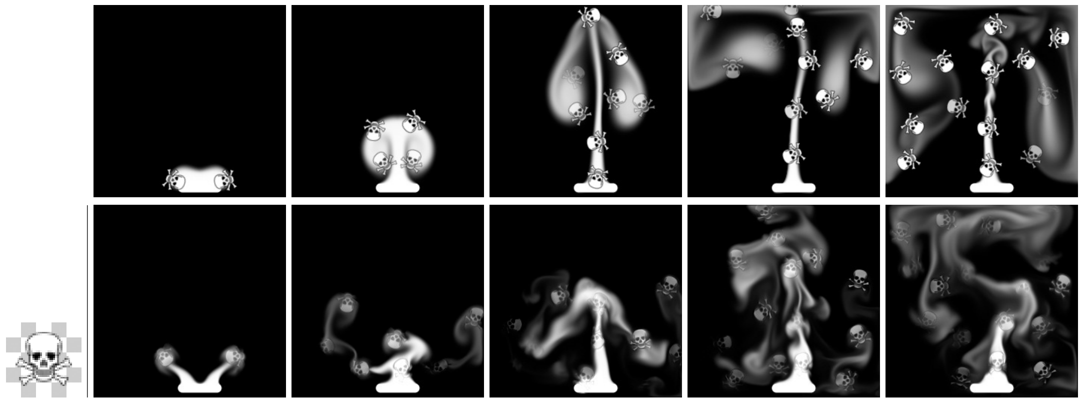


Figure 5: The importance of feedback to the particle control method. Simply advecting motif particles (**top**) produces a far less visually appealing result than using them as a control target (**bottom**).

Here, t_0 is the time at which the particle began fading.

If a particle is found to be matching within tolerance and non-overlapping at any point before it has fully faded out, it is instantly returned to full strength, and no longer marked for removal. In our experience, this did not cause artifacts because motifs dissipate much more easily than they form. However, if a high gathering value were used, it may be desirable to slowly ramp up motif intensity rather than allowing motifs to instantly reach full strength.

4.3 Motif Texture

Whereas motif particles work on a local scale, our motif texture method produces a global effect across the entire fluid. An artist specifies a texture motif as a single sample image of a density field that represents the look and behavior they want to capture. It is the job of our motif texture method to construct a target density field which corresponds to the current density field but is stylized using the artist-supplied sample (Figure 14). The fluid is driven incrementally towards a stylized version of itself at each frame, and is able to reproduce complex textures over an extended number of frames.

4.3.1 Texture Transfer

We generate the control target using the PatchMatch algorithm of Barnes et al. [3], with the current density field as the target and the texture motif as the source. This constructs an image conforming to the current density field, but with the texture of the motif.

PatchMatch works by incrementally refining a dense correspondence of patches between the current density field and the source texture, using a stochastic search. The algorithm attempts to minimize some distance function across the patches which, in our implementation, consist of the 3×3 square area surrounding the destination index and source coordinates, weighted according to a Gaussian function with standard deviation $\frac{1}{2}$.

The correspondence, consisting of offsets and rotation angles from the destination into the source image, is initialized using random but valid values. It is then refined using a two-stage process. First, new, randomly generated offsets and angles are tested at each pixel. The randomization function used to generate the new offsets is weighted such that offsets closer to the current offset are more likely to be tested. Second, offsets along with their corresponding angles are transferred to adjacent pixels, in both forward and reverse scan-line order. This allows large poorly matching areas to be quickly refined using a small number of good matches.

We use a modified distance function in order to capture large-scale features. Instead of simply computing the weighted distance

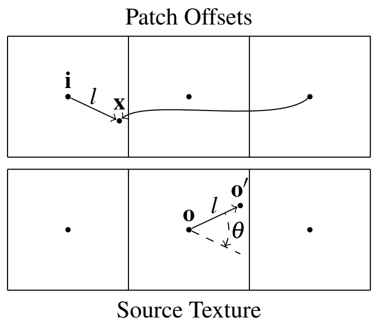


Figure 8: In order to prevent drift in the texture offsets, we must account for the offset between the end-location of the advection backtrace and the nearest grid point (Equation 9).

between pixels at the source and destination, we compute this distance across all levels of both a Gaussian pyramid and a Laplacian pyramid (Figure 15). We then sum the distances over all levels of each, and add these cumulative distances according to a user-specified weighting (Figure 11). This allows us to capture not only the absolute color of the target texture motif, but also relative color changes. By taking the distance at multiple scales, we capture both fine-scale texture and large-scale shapes in the target.

In order to construct the density target from the patch correspondence, we simply compute the average at each pixel of all patch colors with a non-zero weight at that pixel. Given our 3×3 patch size, this means the color at each pixel of the target is controlled by the patch at that pixel, as well as the patches at all surrounding pixels. As with the particle motifs, we do not hope to force the fluid to conform to this target in one frame, but instead rely on the cumulative effect over several frames.

4.3.2 Texture Advection

We do not initialize patch offsets and angles to random values at every frame. Instead, we advect the correspondence from the previous frame. This advection is similar to density advection with several additional requirements. Namely, it does not make sense to interpolate patch offsets or angles, since interpolating between patches which do not correspond to the same area of the source texture will yield an arbitrary new offset and angle which we cannot expect to correspond to a good match. Additionally, patch angles must be updated based on the fluid motion.

The first requirement is easily satisfied by using nearest-neighbor interpolation at the end of the backtrace step. However, this leads

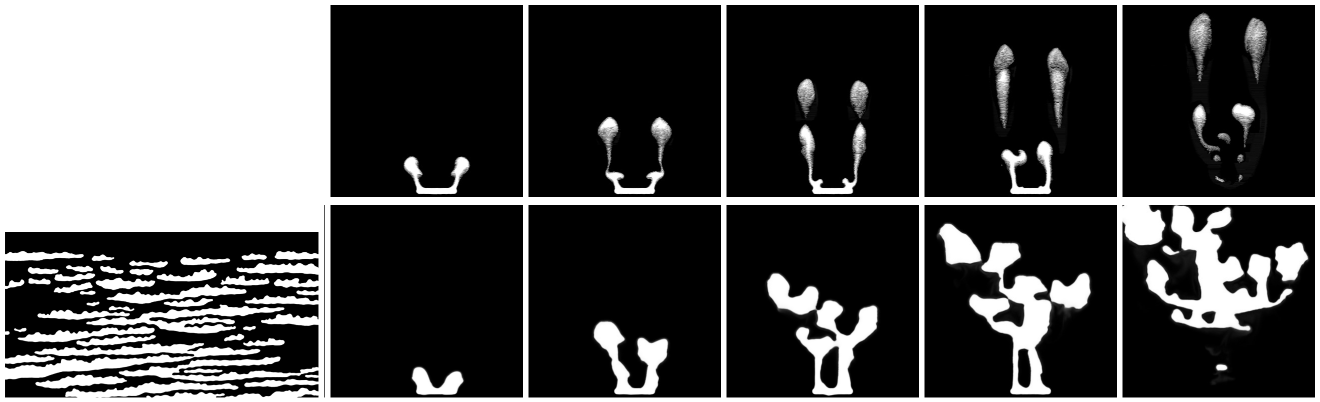
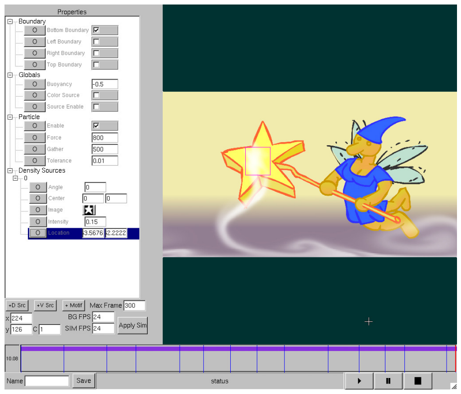


Figure 7: Using PatchMatch-based guided texture synthesis [3] as a stylization primitive. With feedback disabled, the unstylized output (**top**) takes on a bit of the texture of the motif (**left**), but the shape of the fluid remains unchanged. With feedback, the smoke takes on shapes present in the texture (**bottom**).



Properties

- Boundary
  - Bottom Boundary ☑
  - Left Boundary ☐
  - Right Boundary ☐
  - Top Boundary ☐
- Globals
  - Buoyancy -0.5
  - Color Source ☐
  - Source Enable ☐
- Particle
  - Enable ☑
  - Force 800
  - Gather 500
  - Tolerance 0.01
- Density Sources
  - 0
    - Angle 0
    - Center 0 0
    - Intensity 0.15
    - Location 3.5676 2.2222

Max Frame 300, x 224, y 126, C 1, BG FPS 24, SIM FPS 24

Figure 9: Our interface supports rudimentary keyframing of simulation and control parameters.

Grid Size	128^2		256^2		512^2	
	bw	col	bw	col	bw	col
sim	40	41	96	93	254	259
control	56	81	148	247	469	879
16 × 16 particle	70	99	266	383	970	1614
32 × 32 particle	59	80	228	337	905	1450
fast texture	245	343	989	1355	3693	5171
texture	349	490	1436	1790	5039	6773

Table 1: Timings for our system, in milliseconds per frame. Timings are given for both monochromatic (bw) and color (col) simulations. *Sim* is the basic fluid simulator, *control* is our implementation of target-driven smoke, *particle* is our motif particle stylization method (including matching, advection, and validation of motif particles), and *texture* is our texture motif stylization method (including PatchMatch and advection). In the *fast texture* timings, PatchMatch is limited to one iteration and only 10% of the offsets are randomized each frame. Timings we generated using a system with an Intel Core i7 920 CPU.

to a discrepancy between the actual offset we are sampling and the offset to which the value corresponds, which we must account for (Figure 8). In order to satisfy the second requirement, we use the angular difference between velocity at the start and finish of the backtrace, to account for rotation during advection. We adjust the patch offset $\mathbf{o}$ and angle θ as follows:

$$\mathbf{o}' \leftarrow \mathbf{o} + \text{rotate}(\mathbf{x} - \mathbf{i}, -\theta) \tag{9}$$

$$\theta' \leftarrow \text{angle}(\mathbf{v}) - \text{angle}(\mathbf{v}_\circ) \tag{10}$$

Here, $\mathbf{x}$ and $\mathbf{v}$ are the coordinate and velocity at the end of the backtrace, $\mathbf{v}_\circ$ is the velocity at the start of the backtrace, and $\mathbf{i}$ is the coordinate of the nearest pixel to $\mathbf{x}$. angle is a function which computes the angle of a vector relative to the x-axis, i.e. $\text{angle}((x,y)) = \text{atan2}(y,x)$. rotate rotates a vector counterclockwise by some angle, i.e. $\text{rotate}((x,y),\phi) = (r \cdot \cos(a), r \cdot \sin(a))$ where $r = \sqrt{x^2+y^2}$ and $a = \phi + \text{angle}((x,y))$.

The new offset $\mathbf{o}'$ and angle θ' are stored at each location in the new offset field.

5 Interface

Our interface (Figure 9) allows artists to set and keyframe various important features of the fluid simulation (density and velocity emitters, target-driven smoke control parameters, fluid parameters) as well as to specify motifs and their associated parameters (e.g. thresholds for spawning and removing motif particles and Gaussian vs. Laplacian weighting for texture motifs).

The simulation is displayed as an overlay on a user-specified frame-set, providing a rudimentary preview of what the final animation will look like. However, we anticipate that most final animations will be composited in external software.

Our simulator runs at interactive rates, which facilitates artists tuning parameters to achieve their desired goals (Table 1). In the case of the fairy animation, which uses only particle motifs, our simulator runs at almost exactly the animation frame rate. Our simulator is slowed significantly by the use of texture motifs, but we did not find this to be a problem when generating the wizard animation. By working at a lower frame rate and disabling textures when not needed, we were able to work efficiently even when using texture motifs.

6 Results

Our stylization methods produce good results in a variety of situations, including animations combining both particle and texture motifs (Figure 12) and animations containing multiple particle motifs (Figure 13). The feedback mechanism inherent to our technique allows motifs to influence long-term fluid behavior (Figures 5 and 7), and due to our choice in fluid control technique, our system can separate multiple blended colors in a fluid to form complex multi-colored motifs (Figure 16).

Our system was able to achieve our goals for fluid integration

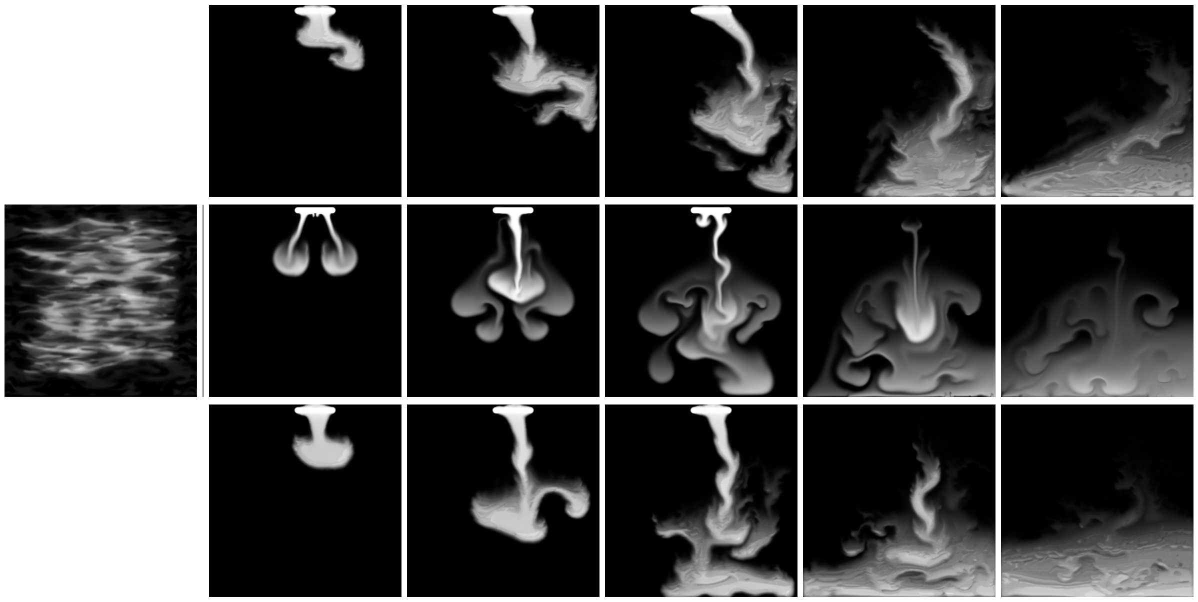


Figure 11: A comparison of different Gaussian vs. Laplacian weightings. **Top**, the Laplacian difference is weighted 0.8 while the Gaussian is weighted 0.2. This somewhat emphasizes the edges present in the texture. **Middle**, an identical underlying simulation without the mist motif. The velocity attenuation step of target-driven smoke remains enabled. **Bottom**, the Laplacian difference is weighted 0.2 and the Gaussian is weighted 0.8. This places more emphasis on matching absolute color values.

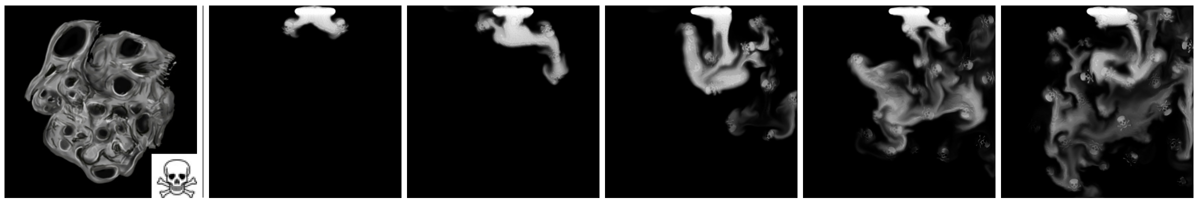


Figure 12: By combining an eerie texture and skull particle, we are able to control both specific motif shapes and overall fluid appearance.

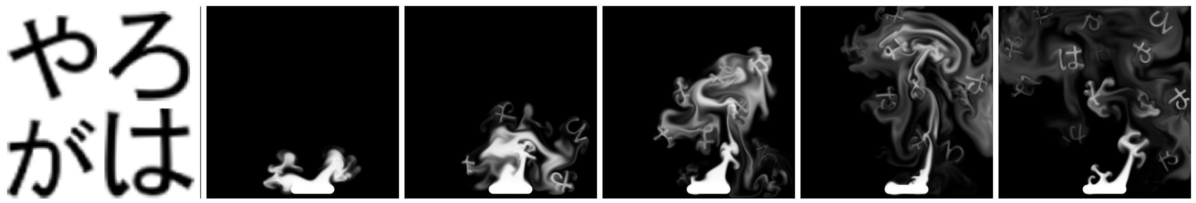


Figure 13: Hiragana characters form from wisps of smoke.

with animations. In the frog animation (Figure 17), we were able to indicate the transition of the pot's contents from a benign brew to a putrid potion by using varying colors of smoke and applying a skull motif. We enhance the effect by enabling a texture motif to give the smoke more substance. In the smoking animation (Figure 10), the character quizzically cocks an eyebrow as she inhales, and her puzzled attitude is reflected by the question-mark motifs that emerge as she exhales. In the wand animation (Figure 1), we indicate the power of the fairy's giant wand through an aura of spirals and stars.

Our simulator is based on the approach outlined by Stam [16], with velocities represented at cell edges instead of centers, added steps to compute smoke control forces and perform gathering, and a multigrid linear solver instead of pure Gauss-Seidel relaxation. In addition, we have used the Cilk++ SDK to parallelize most of our simulator, thus allowing it to take full advantage of modern multicore machines. A table of timings (Table 1) shows that the time taken by our particle stylization approach is comparable to that of the other components of the system, while texture stylization increases running time to roughly 10 times that of our target-driven smoke implementation alone.

When using particle motifs alone, our simulation environment can achieve real-time performance for reasonably large simulation sizes. (The fairy animation was rendered at a resolution of 488×252.) However, slow texture running times are mitigated by our system's automatic and continuous background rendering. Whenever the user makes a change, the animation is re-rendered automatically from that point, with the progress shown in the timeline. In practical terms, this means that the system remains useable even when frame rendering is less than real-time.

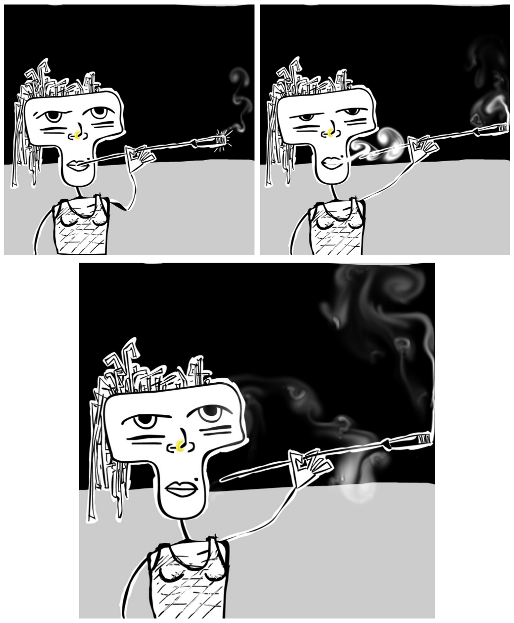


Figure 10: The woman exhales a cloud of quizzical smoke. Smoke is controlled to form question marks.

7 Discussion

In this paper, we introduced the idea of mid-level fluid control – an approach where artists specify desired fluid behavior at a larger scale than basic simulation parameters, but without resorting to global keyframes. To make this notion concrete, we implemented a mid-level smoke controller which causes the fluid to conform to artist-specified density motifs. With the help of our user interface, fluids controlled in this way can be integrated into two-dimensional animations to achieve a variety of artistic goals.

Our controller is built around the idea of feeding back stylization information through global pose control, a basic blueprint that may prove useful in future mid-level control schemes. We expect that many techniques used to stylize a fluid animation as a post-processing step could benefit from this feedback mechanism.

Indeed, while we have focused on density, it would be interesting to also specify *velocity motifs*, to control patterns of flow, and *surface motifs*, to control boundary shapes when working with liquids. In the extreme, one could consider simulating a fluid over a fully user-specified basis, perhaps consisting of translated and rotated copies of density and velocity motifs. In this case, mid-level control could be as simple as encouraging sparsity in the representation – in other words, driving the fluid to states that can be represented with just a few basis elements.

While our current motif-particle stylization approach is limited to 2D fluids, we see two routes to generalizing it to 3D: The technically simplest generalization would be to use volumetric motifs. In this case the matching and update steps change trivially, both in the case of texture and particle motifs (take distances at every voxel instead of every pixel); however, the artistic effort required to create volumetric motifs is likely to be substantial, and it will be hard to guarantee that the camera will always view a given motif from a pleasing or meaningful angle. Another, more promising, option is to use 2D, screen-space motifs. In this case, particle matching and updates could be performed either on the projection of the whole fluid volume or a slice thereof to screen space. However, this approach requires a new view-dependent fluid control method which may be technically difficult to concoct. Nonetheless, both approaches are interesting future work.

Though we do expose intrinsic fluid parameters to the artist for adjustment, it could be useful to automatically select a set of fluid parameters that suit the user-specified motifs. For instance, a very swirly motif would tend to imply a low-viscosity fluid, while a smooth motif would be natural in a very diffusion-prone setting.

In the future, mid-level fluid control methods have the potential to be a powerful tool for artists who wish to direct the general look of fluid – velocities, densities, surface shapes, and the time-evolution thereof – without having to specify the exact global fluid configuration. Mid-level control is a cooperation between man and machine, integrating the stylistic vision of artists with the computer's grasp of fluid dynamics; producing results that neither party could create alone.

References

[1] A. Angelidis, F. Neyret, K. Singh, and D. Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 25–32, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[2] A. W. Bargteil, F. Sin, J. E. Michaels, T. G. Goktekin, and J. F. O'Brien. A texture synthesis method for liquid animations. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009.

[4] C.-H. C. Du-Ming Tsai. Rotation-invariant pattern matching using wavelet decomposition. In *Pattern Recognition Letters*, pages 191–201. Elsevier Science Inc., January 2002.

[5] R. Fattal and D. Lischinski. Target-driven smoke animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 441–448, New York, NY, USA, 2004. ACM.

[6] N. Foster and D. Metaxas. Controlling fluid animation. In *CGI '97: Proceedings of the 1997 Conference on Computer Graphics International*, page 178, Washington, DC, USA, 1997. IEEE Computer Society.

[7] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson, and M. Lin. Texturing fluids. *IEEE Trans. Visualization and Computer Graphics*, 13(5):939–952, 2007.

[8] C. Ma, L.-Y. Wei, B. Guo, and K. Zhou. Motion field texture synthesis. *ACM Trans. Graph.*, 28:110:1–110:8, December 2009.

[9] M. McGuire and A. Fein. Real-time cartoon rendering of smoke and clouds. In *NPAR*, June 2006.

[10] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 449–456, New York, NY, USA, 2004. ACM.

[11] F. Neyret. Advected textures. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*, July 2003.

[12] M. B. Nielsen and B. B. Christensen. Improved variational guiding of smoke animations. *Comput. Graph. Forum*, 29:705–712, May 2010.

[13] J. Schpok, W. Dwyer, and D. S. Ebert. Modeling and animating gases with simulation features. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–105, New York, NY, USA, 2005. ACM.

[14] A. Selle, A. Mohr, and S. Chenney. Cartoon rendering of smoke animations. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 57–60, New York, NY, USA, 2004. ACM.

[15] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Trans. Graph.*, 24(1):140–164, 2005.

[16] J. Stam. Real-time fluid dynamics for games. In *Proceedings fo the Game Developer Conference*, march 2003.

[17] J. Tan and X. Yang. Physically-based fluid animation: A survey. *Science in China Series F-Information Sciences*, 52(5):723–740, 2009.

[18] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 716–723, New York, NY, USA, 2003. ACM.

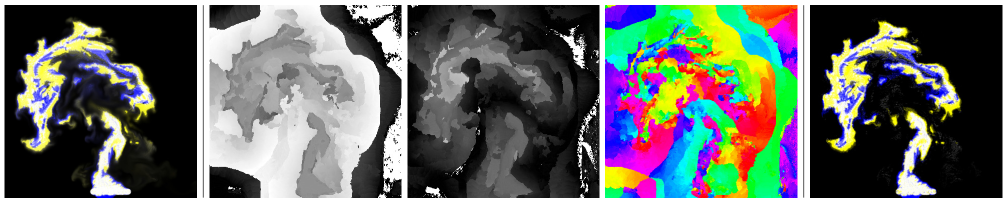


Figure 14: **Left**, a frame from an animation using our motif texture method. **Middle**, from left to right: patch x offsets, patch y offsets, and patch rotation angle (red is $0°$, green is $120°$, blue is $240°$). **Right**, the stylized density field (used as the control target for the next frame).

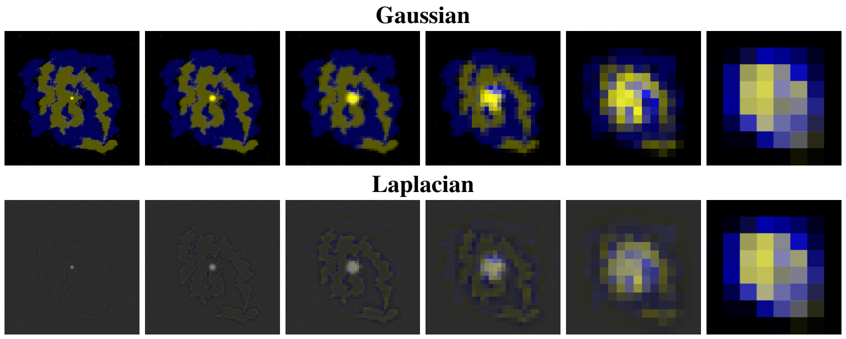


Figure 15: We use multi-resolution matching on both Gaussian and Laplacian pyramids in order to capture both the absolute color and edges of the target motif, with a user-specified weighting between the two. A sample Gaussian filter kernel is overlaid on each image, to show which portions might contribute to a single patch distance computation.

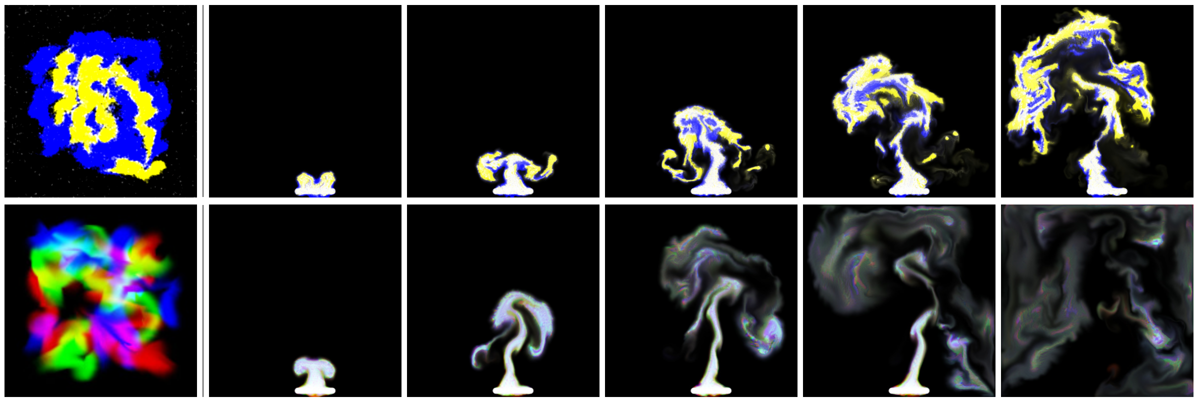


Figure 16: The gathering term of the target-driven smoke control technique [5] allows motifs to separate white smoke into its components.

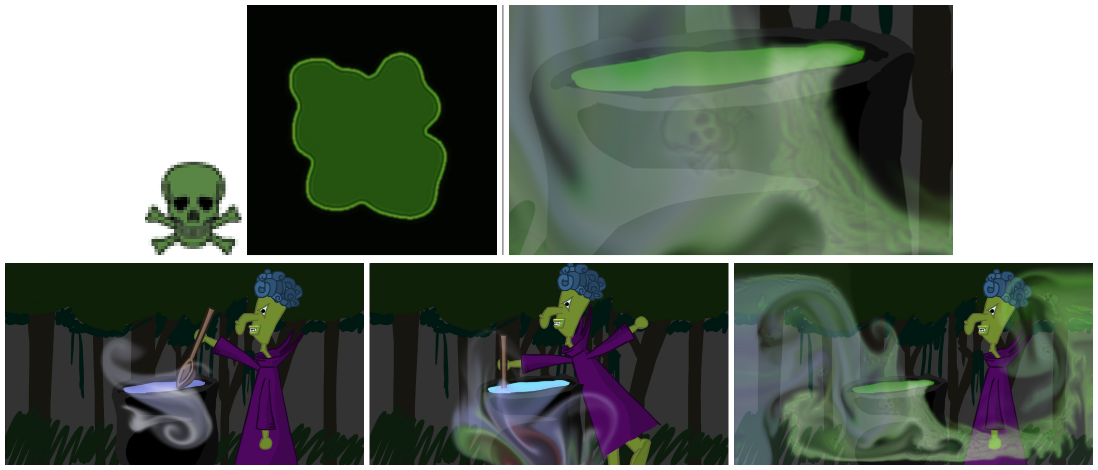


Figure 17: **Top right**, a larger view from the last frame below highlights the motifs shown on the **top left**. **Bottom**, a wizard puts the final touches on a toxic potion. Smoke is controlled to form skull motifs and green borders.