Pacific Graphics 2011
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin
(Guest Editors)

Volume 30 (2011), Number 7

Creating Fluid Animation from a Single Image using Video Database

Makoto Okabe[†1,2] Ken Anjyo[‡3] Rikio Onai[§1]

[1]The University of Electro-Communications [2]JST PRESTO [3]OLM Digital, Inc. / JST CREST

Abstract

We present a method for synthesizing fluid animation from a single image, using a fluid video database. The user inputs a target painting or photograph of a fluid scene along with its alpha matte that extracts the fluid region of interest in the scene. Our approach allows the user to generate a fluid animation from the input image and to enter a few additional commands about fluid orientation or speed. Employing the database of fluid examples, the core algorithm in our method then automatically assigns fluid videos for each part of the target image. Our method can therefore deal with various paintings and photographs of a river, waterfall, fire, and smoke. The resulting animations demonstrate that our method is more powerful and efficient than our prior work.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Motion

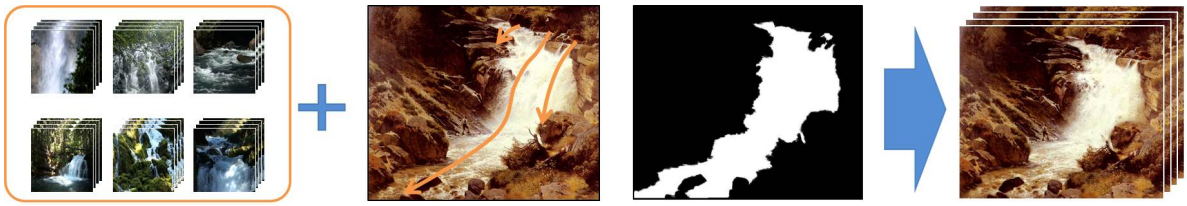
1. Introduction

Pictures were first animated in lift-the-flap books, and the animation of pictures is now recognized as a classic visual effect in the animation industry. It is also an active area of research within the field of computer graphics [HAA97, IMH05, HDK07]. In the animation of pictures, the designer specifies a single target image along with several characteristics regarding motion and uses a computer to synthesize animated sequences derived from the input. Of course, the level of difficulty involved in animating a picture varies markedly according to the complexity of the scene and the objects to be animated. It is difficult to animate a picture of a fluid. Early research [CGZ*05, OAIS09] was successful in designing fluid animation. Here, we focus on this challenging issue using an efficient data-driven approach to handle a wide variety of fluid motions that could not be animated previously.

A picture of a fluid can be animated in three ways. First, a physics-based fluid simulation can be applied to the fluid part of a target image. However, it is difficult to set the many physical parameters required to reproduce the target appearance. The second method can synthesize relatively calm fluid motions, such as water surfaces [CGZ*05]. However, our study focuses on synthesizing more dynamic motions, such as water splashes, smoke, or fire, in addition to calm fluids. A third method, which we described in our previous paper, requires users to specify a video and then transfers its fluid features to the target image [OAIS09]. This technique relies on a single video example that limits variation in the available fluid features. Another problem is that the user-specified motion field is temporally stationary, which limits the dynamics. The user must also expend considerable effort to find an appropriate video example and specify the motion field.

To address these problems, we develop a data-driven method to create a fluid animation from a picture (Fig. 1). The user inputs a target image and an alpha matte that extracts the fluid region of interest, while providing a few optional suggestions about fluid motion (i.e., flow direction and speed). We employ a video database that includes hundreds of video examples of fluids, which therefore helps the user to synthesize better quality animations with less effort than with previous methods. The quality of the synthesized animation is improved due to the large variety of available video examples. We test the flexibility of our method by synthesiz-

† m.o@acm.org
‡ anjyo@olm.co.jp
§ onai@onailab.com



(a) Video Database (b) Target Image (c) Alpha Matte (d) Synthesized Animation

Figure 1: *We employ a database of video examples of fluids (a). The user specifies a target image (b) with a few optional suggestions about fluid motion, e.g., sketches of flow direction, shown as orange arrows. The user also provides an alpha matte of the region of interest (c). The system synthesizes an animation (d).*

ing a wide variety of fluid animations, i.e., river, waterfall, smoke, and fire.

The present study makes in three major technical contributions to solve the problem addressed in this paper. First, we develop an image-search-based technique to efficiently extract local fluid features from a video database. We then cut each video into smaller pieces, which are vector-quantized to construct a bag-of-features codebook for efficiently finding a video piece with an appearance that is similar to part of the picture. Second, we develop an algorithm to assign the video pieces to the picture so that the integrated appearance and motion become smooth. This task is formulated as a multi-label assignment problem based on the Markov random field (MRF). The third contribution is the extension of the original synthesis algorithm. In our previous study, we decomposed a video into three components, i.e., the average image, the motion field, and the residuals [OAIS09]. However, we found that two components (the average image and the differences between the average image and the original frames) produce a wider variety of dynamics than previously possible by using the video database effectively. Additionally, our approach markedly reduces the user burden, as illustrated below.

2. Previous Work

Image and Video Database Image and video searching is an active research field. Recently, this kind of search has also been applied to image and video syntheses for computer graphics applications. Millions of images are useful for scene completion [HE07]. Sketch2Photo allows a user to draw rough sketches and search for adequate images from a database to synthesize a desired scene [CCT*09]. Skyfinder searches for a user-desired sky image, modifies its appearance, and creates a composite [TYS09]. With regard to video synthesis, SIFT Flow can be used to estimate the motion field in a single image and also to transfer part of a video example onto a single image [LYT*08]. Webcam Clip Art provides a video database of outdoor scenes and is useful for scene relighting [LEN09]. However, no existing method addresses the problem of animating a picture of fluid using a video database.

Animating Pictures Many methods have been proposed for creating an animation from a single image [HAA97, IMH05, HDK07]. These methods are useful for touring into a picture and creating character animations, but fluid animation is beyond their scope. Chuang et al. proposed a method to synthesize an animation with stochastic motion from a single image [CGZ*05], but this technique only supports oscillation of the water surface, such as ripples, and not flowing fluid animation. Lin et al. synthesized animation from multiple, instead of single, high-resolution stills [LWW*07].

Video Texture Synthesis Video texture synthesis has also been well-explored [WL00, BJEYLW01, SSSE00, DCWS03]. However, it is difficult for users to modify the appearance or motion of a synthesized animation. Wang and Zhu analyzed fluid animation, represented it with textons, and synthesized an animation [WZ03]. Bhat et al. proposed a sketching interface that enables users to edit a fluid animation of a video example [BSHK04]. Users can also change the appearance of the animation, but the problem of animating a picture of fluid has not yet been solved. Kwatra et al. developed a method that can design the animation of a texture flowing over a user-specified motion field [KEBK05], where only a stationary motion field is demonstrated without any high-frequency fluid features. Ma et al. extended example-based texture synthesis to allow an exemplar to affect the details of motion fields [MWGZ09]. These methods enable users to modify an existing fluid animation and synthesize an animation, but they do not allow users to specify a single image as an appearance constraint. Okabe et al. developed a technique for animating a picture of fluid [OAIS09]; this technique requires users to design a motion field manually and to search for an appropriate video example. We extended this work, incorporating a video database to reduce user effort and to improve the quality of the synthesized animations.

3. System Overview

Given a target image of fluid that a user wants to create a fluid animation, our process involves selecting appropriate video examples from a database, assigning them onto parts of the target image, and integrating them all seamlessly into

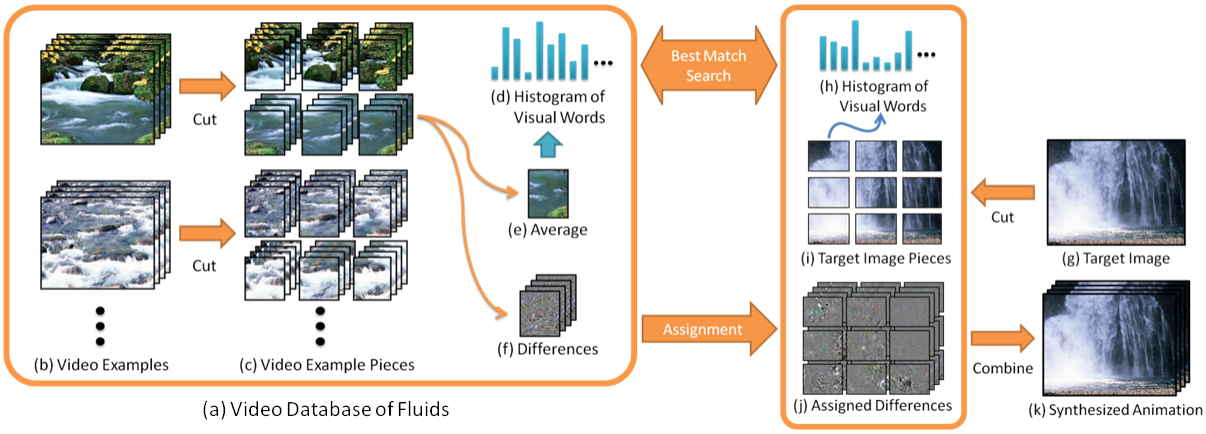


Figure 2: *System overview.*

a final animation. We assume that the user inputs a single target image and an alpha matte that extracts the fluid region of interest. The user can also provide a desired motion field by sketching the flow direction and painting a speed map, which is used as a constraint in the assignment process.

Our system consists of three components: 1) construction of a video database of fluids (Fig. 2-a), where each video example is cut into small pieces, 2) a best-match search for an appropriate video example piece and assignment of this to a part of the target image, and 3) synthesis of the final animation through seamless integration of all the assigned pieces and adjustment of the overall appearance. The offline process of database construction begins with gathering original video examples of fluids (Fig. 2-b). To increase the number of video examples, we cut each video example into smaller pieces (Fig. 2-c). For each video example, we compute the average image by averaging the frames (Fig. 2-e) to obtain representative appearance information. We also compute differences between the average image and the frames (Fig. 2-f) that have no significant color properties but capture high-frequency fluid features. From all of the averaged images in the database, we construct a bag-of-features codebook and describe each average image using a histogram of visual words (Fig. 2-d).

Given a target image (Fig. 2-g), we cut it into pieces using the same process used in the database construction (Fig. 2-i). Next, we compute the histogram of visual words for each piece (Fig. 2-h), perform a best-match search between histograms of visual words (see Figs. 2-d and h), and assign video example pieces that have appearances similar to target image pieces. When a user-specified motion field is given, it is used as a constraint for solving the assignment problem. Based on the assignment results, differences are copied onto the corresponding target image pieces (Fig. 2-j). Finally, all assigned differences are integrated seamlessly and the final appearance is synthesized by adjusting the appearance (Fig. 2-k).

Our data-driven transfer of fluid features to a target image was inspired by *Image Analogies* [HJO*01]; the transferred differences relate to the target image piece in the same way as the differences originally related to the average image of the video example. Our method differs from Image Analogies in that it is not pixel-based but rather is a patch-based texture synthesis, similar to *Image Quilting* [EF01]. When integrating patches, Image Quilting computes a minimum error boundary cut to remove seams between patches; we compute alpha blending using a computationally inexpensive image pyramid. A video example piece is decomposed into its average image and differences; the former captures the overall low-frequency appearance and the latter has high-frequency fluid features. The high-frequency features can be integrated seamlessly through simple alpha blending.

4. Database of Video Examples

We need to prepare as many video examples as possible, because a large amount of data is always important to achieve high-quality data-driven image synthesis [HE07, TYS09]. However, despite the large volume of video data available on the Internet, we found it difficult to gather the tens of thousands of video examples suitable for our purpose. Our criteria for video examples are:

- The camera be fixed and focused on the fluid itself, i.e., the fluid must be the primary character of the video.
- The video examples of fluids need to be of sufficiently high quality for animation synthesis.
- To simplify our method, we use only video examples with no other significant moving object other than fluids.

We gathered several hundred video examples of water scenes from commercially available video footage collections, but discarded any that did not satisfy our requirements. Finally,

we have a total of 151 video examples of water scenes (Fig. 3). All video examples in the database have a resolution between 640×480 pixels and 400×300 pixels.

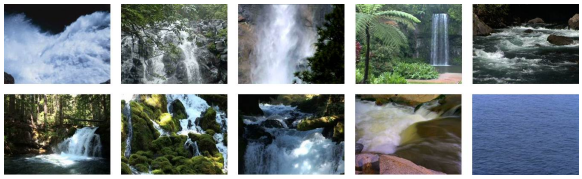
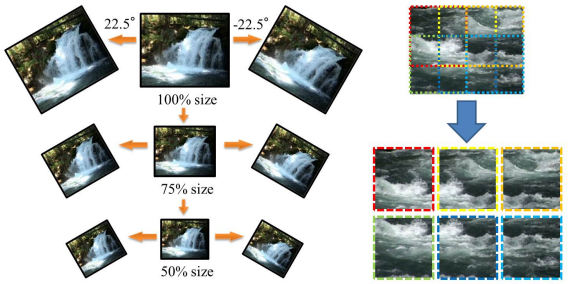


Figure 3: *Thumbnails of video examples in the database. All video examples satisfy three requirements: 1) the fluid is the main character in each scene, 2) each video example has a resolution quality between* 640×480 *pixels and* 400×300 *pixels, and 3) no other moving objects appear in the scene.*

To increase the amount of data, our strategy is to observe each video example locally. For example, a video example of a waterfall may have a different global shape from the shape of the waterfall in the target image. However, its local parts, such as droplets or splashes, may be used to animate a part of the target image.

To increase the number of video examples in the database, we flip each video example horizontally (thereby doubling the number of examples); we also scale and rotate each example (Fig. 4-a). We then cut examples into smaller video pieces (Figs. 2-c and Fig. 4-b). We make three versions of each video example by scaling them to 50, 75, and 100% of their original size. We also make three versions of each video example by rotating the original by -22.5, 0.0, and $+22.5$ degrees. Each version is then cut into smaller pieces with a resolution of 48×48 pixels, allowing neighboring pieces to overlap each other (Fig. 4-b). Such an idea of patch library is also used for synthesizing facial image [MPK09].



(a) Scaling and rotating a video example. (b) Video example pieces.

Figure 4: *Database construction.*

Video example pieces that include not only fluids but also other objects such as rocks, trees, or the sky degrade the efficiency of the database. Therefore, we remove such stationary examples by calculating the significance of motion. For each piece, we compute optical flow between neighboring frames (Fig. 5-b) to obtain an average through all of the frames (Fig. 5-c). In a video example, only a pixel position with a significant averaged motion is used; other pixels are masked out (Fig. 5-d). In addition, any video example in which no pixel position has significant motion is completely removed from the database. We use *OFLib* to compute a dense optical flow [ZPB07]. This algorithm has three important open parameters: θ, λ, and the number of iterations, which we set to 1.0, 0.8, and 25, respectively. Because a fluid video has high-frequency dynamics, we set parameters that reduce sensitivity to such dynamics but still yield relatively smooth results. We have collected 246,477 video examples of water scenes.

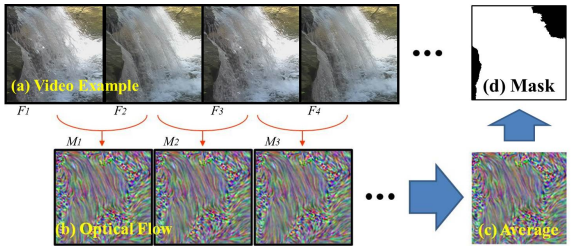


Figure 5: *Processing a video example piece.*

To ensure efficient best-match searching and efficient assignment of video examples, we construct a bag-of-features codebook, by which each video example piece is described using a histogram of visual words (Fig. 2-d). The bag-of-features technique works well for searches involving small image patches such as those in our database, especially when it is combined with spatial pyramid representation [SSSFF09]: while bag-of-features has only information of presence of textures, the spatial pyramid representation adds information of location of textures that is important even for our small patches. We make a representative image by averaging the frames of each video example piece when constructing the codebook (Fig. 2-e). Averaging involves the motion blur that often appears in a picture of fluid; this motion is often intentionally portrayed by a painter or a photographer to visualize the trajectories of fluid motion. We extract SIFT features of each representative image by applying a SIFT descriptor, and compute 128 dimensional feature vectors at each point on a regular grid of 9×9 over the 48×48 pixel resolution image. The use of SIFT descriptors on a regular grid is often better for image searching than using key points detected by Gaussian differences [LP05].

We then compute vector quantization on all extracted SIFT features to obtain the visual words used to construct the codebook. For the vector quantization, we apply a repeated cluster bisectioning approach, because this method is reportedly faster and yields a better quality of clustering than K-means [ZKF05]. Similarly to K-means, we set the number of clusters, k, based on experimental results for settings of 100, 200, 300, and 500; we have chosen 200 as the best parameter. Given the codebook, we generate a bag of features for each representative image of a video example piece. We assign

each SIFT feature to its nearest visual word and construct a histogram with spatial pyramid representation [LSP06].

5. Assignment of Video Example Pieces

Given a target image, we first decompose it into pieces (Fig. 2-i) in the same manner as in the database construction (Fig. 4-b). Next, we assign an appropriate video example piece to each target image piece. Our solution of the assignment problem involves three criteria: 1) appearance similarity between a video example piece and a target image piece; 2) smoothness of motion fields between neighboring assigned pieces; and 3) appearance similarity between neighboring assigned pieces. The first criterion means that the more similar a video example piece is to a target image piece, the more likely it is that both pieces have similar fluid motions. The second criterion means that if a part of the target image is assigned a fluid motion, its neighboring piece is likely to have a similar fluid motion, i.e., their velocities and temporal video textural patterns would be similar. The third criterion prevents neighboring pieces from generating an inconsistent appearance in the video texture.

To address the first criterion, we compute appearance similarity by comparing bag-of-features histograms (Figs. 2-d and h). As in the database construction, we extract SIFT features on the regular grid in each target image piece and describe the image feature as a histogram of visual words. Appearance similarity is computed using a histogram intersection,

$$I(H_e, H_t) = \sum_{i=1}^{k} min(H_e[i], H_t[i]), \quad (1)$$

where H_e and H_t are the histograms of visual words of a video example piece and a target image piece, respectively.

For the second criterion, we assume that the similarity measure of the average motion field is a simple definition of smoothness between neighboring motion fields (Fig. 5-c). However, this definition is insufficient, as shown in Fig. 6, when each video example piece has a similar average motion field; that is, when they all flow from top to bottom at a similar speed. However, if we select motion fields and assign them to neighboring target image pieces, the result looks unnatural because their temporal video textural patterns differ. Fig. 6-a shows a small waterfall captured from a short distance in which the motion is temporally unstable due to visible droplets and splashes. Fig. 6-b , however, shows a large waterfall captured from a long distance, with a temporally smooth motion. These differences in temporal patterns are well described by Fourier analysis; Fig. 6-a has uniformly strong coefficients at all frequencies whereas Fig. 6-b has strong coefficients only in the DC component and weak coefficients at other frequencies. We can compute the Fourier transform at each pixel position x as $\hat{M}(x) = \mathcal{F}(M(x))$. $\mathcal{F}$ is the one-dimensional discrete Fourier transform and conversion to the power spectrum; $M(x)$ is a sequence of motion field at x, i.e., $[M_1(x), M_2(x), ..., , M_N(x)]$ and N is the number of frames. We can thus define the smoothness of neighboring motion fields, M^i and M^j, as

$$smooth(M^i, M^j) = d(\hat{M}^i{}_1, \hat{M}^j{}_1) + \sigma \sum_{f=2}^{N/2} d(\hat{M}^i{}_f, \hat{M}^j{}_f), \quad (2)$$

$$d(F, G) = \sum_{x \in \Omega} (F(x) - G(x))^2, \quad (3)$$

where Ω is a set of pixel positions in the overlapping area between neighboring target image pieces, F and G, and σ is a parameter balancing the DC component, which corresponds to the average motion field, and the AC component, which describes the higher-frequency dynamics.

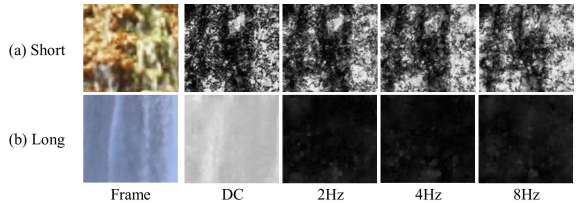


Figure 6: *Fourier analysis on motion fields.*

For the third criterion, we can describe the textureness of an assigned video example piece and measure its similarity to neighboring pieces. Fig. 7 illustrates why this criterion is required. The two video examples of smoke have different appearances: one has strong contrast and the other has a smooth appearance. However, because they have similar motions, flowing from left to right, the first and second criteria allow them to be neighboring pieces and will result in a visual artifact with inconsistent neighboring textures: one has a high contrast appearance and its neighbor has a low contrast appearance. This is because the first criterion is based on a bag of features of a SIFT descriptor, which is invariant to illumination and contrast change and therefore too weak to preserve consistency. To solve this problem, we can describe the textureness of each frame of a video example piece using a four-level scale-space Laplacian pyramid, and sum the texture through all frames; this process distinguishes Fig. 7-a and b, because the former has stronger coefficients than the latter through all of the sub-bands. The textureness of each video example piece, T, has a dimension of $48 \times 48 \times 4$. The similarity of neighboring textureness, $texture(T^i, T^j)$ is computed as the Euclidean distance in the overlapped area, similar to Eq. 3.

We solve the assignment problem by first selecting multiple video candidates for each target image piece based on their similar appearance (Eq. 1). We set the number of candidates to 100, i.e., each target image piece has 100 labels, then determine which label should be assigned to each target image piece. We next formulate the assignment problem with all of the criteria using MRF, which can be expressed

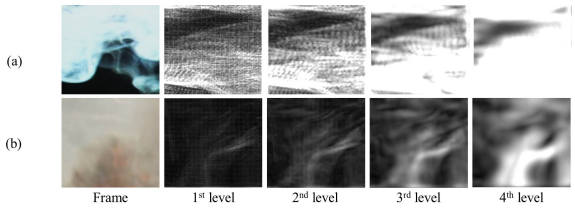


Figure 7: *Analyzing textureness using Laplacian pyramid.*

as:

$$\arg\min_i E = \sum_p V_p(l_i) + \lambda \sum_{(p,q)} W_{p,q}(l_i, l_j), \quad (4)$$

where labels, l_i and l_j, are assigned to the neighboring target image pieces, p and q. V_p is the data term and $W_{p,q}$ is the smoothness term; these are defined using a balancing parameter ρ, as follows:

$$V_p(l_i) = I(H_p, H_{l_i}), \quad (5)$$

$$W_{p,q}(l_i, l_j) = smooth(M^{l_i}, M^{l_j}) + \rho \cdot texture(T^{l_i}, T^{l_j}). \quad (6)$$

To ensure energy minimization, we apply the graph-cut method with a-expansion [BVZ01], which is provided as a package by [SZS*08]. After the graph cut is run, each target image piece has an assigned video example piece that best satisfies all criteria.

User-Specified Motion Field A user can manually specify a desired motion field and influence the assignment process, which is useful for controlling dynamics or modifying a failed assignment. To make the editing process as easy as possible, we can decompose motion field into orientation and speed maps [OAIS09]. The user draws using strokes (Fig. 8-a) and generates an orientation map by interpolating the sparsely drawn strokes using a radial basis function (Fig. 8-b). The speed map is a gray-scale image that is easily edited using paint software (Fig. 8-c). The user can specify the orientation and the speed map, or only one of these. It is possible to edit a speed map from scratch, but it is easier to edit a speed map obtained from the already assigned video example pieces. The edited motion field affects the selection of the 100 candidates: before the selection, we can narrow down candidates in advance by discarding video example pieces with motions that differ greatly from the user-specified motion field.

6. Video Synthesis

Integration of Assigned Video Example Pieces After assignment, we expect neighboring target video example image pieces will have video textures. However, because they are likely to come from different parts of different video examples, naïve integration of these pieces causes visible artifacts, especially discontinuities along the boundaries between neighboring pieces. Our strategy for seamless integration is to decompose a video example piece into its average image and differences. We average all the frames of each video example piece as $A = \frac{1}{N}\sum_{i=1}^{N} F_i$, where F_i are the frames of the video example piece and N is the number of frames. We compute differences as $D_i = F_i - A$. These differences capture high-frequency fluid features, which can be integrated using alpha blending without introducing artifacts, and not the low-frequency appearance (Fig. 9). To merge smoothly overlapped areas between neighboring target image pieces, we apply an image pyramid [BA83] and perform alpha blending for each sub-band. This generates better results than standard alpha blending along boundaries because it can introduce more continuity at low frequencies and also preserves high-frequency fluid features.

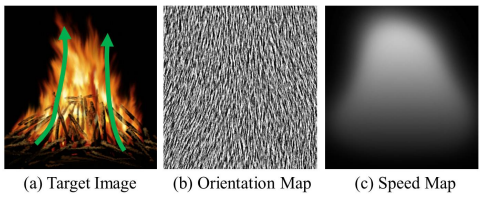


Figure 8: *User-specified motion field. Here, the user wants fire to move from bottom to top along the green arrows (a and b) with a slower speed at the bottom and a higher speed at the top (c).*

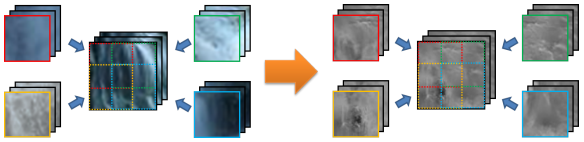


Figure 9: *Integration of neighboring video example pieces. The left side illustrates how each target image piece has an assigned video example piece, and the right side illustrates how corresponding differences are copied and integrated using alpha blending with an image pyramid.*

Recovery of Color Appearance The integrated differences have video textures but no color appearance. To recover the color of the target image, we can synthesize an image that corresponds to the average image (Fig. 2-e) and add it onto the integrated differences. We can synthesize this approximated average image by applying image-based motion blur [BE01] and Gaussian blur to the target image. For motion blur, we compute a motion field over the target image, based on the motion fields of assigned video pieces. An open parameter to simulate shutter speed controls the effect of motion blur. The additional Gaussian blur removes the sharp edges that are introduced by image-based motion blur; kernel size is determined in proportion to the magnitude of the motion field.

Appearance Matching Finally, we can match the appearance of the synthesized animation more precisely to the

target image. We apply histogram-matching between each frame of the synthesized animation and the target image in the same manner as [OAIS09] using Heeger and Bergen's texture synthesis algorithm [HB95]. To recover local appearance, we divide the image space into regular grids and apply the process to each region independently.

Dynamic Boundaries of Fluid The synthesis method described above is useful for rendering fluid features with a static overall shape, e.g., a waterfall seen from a long distance. However, when we apply this method to fire or smoke, the synthesized animation looks unnatural. We develop a simple ad-hoc method to achieve dynamic boundaries of fluid by advecting the alpha matte according to motion fields [BNTS07]. We can define an image-warping function $W_i(I)$ that deforms an image I according to the motion field M_i of the synthesized animation. Given an alpha matte α, we repeatedly synthesize the time-varying alpha matte B_i: $B_{i+1} = \tau W_i(B_i) + (1-\tau)\alpha$, where B_1 is equal to α and τ controls the dynamics of B_i. We also apply an area-preserving adjustment each time B_i is computed to ensure that the area of B_i is always the same. Synthesis of dynamic boundaries requires that the user perform an additional task: manual design of the background image. The background is hidden by fluid in the original image, but it appears when the fluid moves. In our experiment, we used Adobe Photoshop's clone stamp tool and found that it took less than 10 min to design each background.

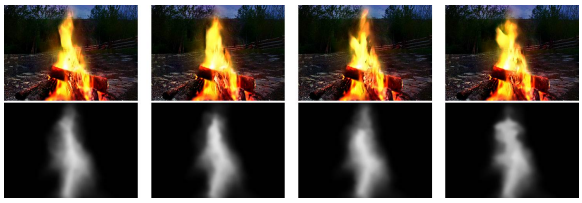


Figure 10: *Each frame of a fire animation has dynamic boundaries and a corresponding alpha matte.*

7. Results and Discussion

We employed an independent database for water, fire, and smoke scenes. This involved gathering 151, 96, and 89 video examples for the water, fire, and smoke databases, respectively. From these, we obtained 246,477, 226,986 and 195,318 video example pieces. We synthesized fluid animations for target images, including photographs and paintings (Figs. 1 and 11). Figure 1 shows the application of our technique to a *tour into the picture* in the supplementary video [HAA97]. We designed an alpha matte for each target image using a scribble-based image segmentation tool [LSTS04]; this process takes less than 5 min. We specified the orientation map for all target images; designing the orientation map requires a sparse set of user-drawn strokes, which is a simple task and takes less than 1 min. We specified the speed map only for the examples shown in Figures. 1, 11-d, and 11-g. Designing a speed map from scratch is difficult, but when synthesizing the first version of an animation, our system outputs the resulting speed map. This is a gray-scale image that can be easily loaded into any paint software. By editing the map, the user can make the animation run faster or slower.

Figure 11: *Thumbnails of synthesized animations: photographs(a,b,c,e,h) and paintings (d,f,g).*

Assignment of Video Example Pieces The supplementary video demonstrates that our assignment algorithm works well, using Figures 11-e and 11-f as target images. Renderings from the assigned video example pieces shown in the supplementary video already resemble the target image. In Figure 11-e, the top region has strong flames, with smaller flames appearing around the tree trunks. For each region, the assignment process copies and pastes appropriate animations from various video examples. Figure 11-f has regions with waterfalls, whitewater, and calm water surfaces; each of these areas is based on appropriate video examples. Waterfalls or regions with flames tend to have successful assignment even without any user input because they have strong image edges that correlate to the flow direction. For example, waterfalls and flames always move from top to bottom and from bottom to top, respectively. High-frequency regions tend to have slower, more detailed motion. However, the flow direction of the water surface in Figures 11-a and 11-h is ambiguous. It is also difficult to determine the flow direction of smoke, because image edges do not correlate to an underlying flow direction and speed. The correlation between appearance and motion is most noticeably absent in paintings. In these cases, orientation and speed maps become especially important.

Timing In our current implementation, which is not optimized, we spend an average of approximately 1 hour creating the first version of the synthesized animation. We use a workstation with an *Intel(R) Xeon(R) 3.33 GHz* cpu for video synthesis and a remote file server for the database. We spend 30 min on the assignment process and 30 min on video synthesis. The assignment process is slowed not by the energy minimization process but by computation of the smoothness term (Eq. 6). Each node of the MRF has 100 candidates, i.e., there are 100×100 edges between neighboring nodes. Computation of each edge requires loading two sets of motion data from the database, and disk access

becomes a bottleneck. A more efficient disk-access process will be developed in future work. We plan to assemble currently fragmented files into a single large file to eliminate the random-seek latency [KSH04].

Open parameters Assignment of video example pieces involves three open parameters: σ to balance the DC and higher frequency components in Eq. 2; λ to balance the data and smoothness terms in Eq. 4; and ρ to balance the smoothness of motion fields and similarity of neighboring textureness in Eq. 6. In our experiment, λ does not significantly affect results, probably because we rely on 100 video example candidates that have already satisfied the data terms to some extent. σ has a significant effect on the results; an overly large value of σ ignores the DC component, which is equivalent to the average motion field, and thus, the resulting animation tends to have chaotic flow directions. ρ is also important, but it has a weaker effect than the smoothness of motion fields. During the experimental process, we set $\lambda = 0.01$, $\sigma = 100$ or 1000, and $\rho = \sigma \times 100$. To reduce the computational time, multiple computers are useful to find the best parameter set. We routinely prepare a variety of parameter sets and run them in parallel.

Duration of Synthesized Animation The duration of the synthesized animation depends on the length of the video examples. However, several methods are available to synthesize an infinitely flowing fluid animation from a video example [SSSE00, DCWS03, BSHK04]. By applying one of these techniques to our database and defining the rule of infinite repetition for each video example, our synthesized animation can also permit infinite repetition.

7.1. Comparison with Previous Methods

It is possible to synthesize fluid animation using a texture-optimization method [KEBK05]. However, this method is not suitable to synthesize time-varying high-frequency fluid features, since it is designed to preserve the original appearance of a given texture image for all frames.

The difference between our method and our prior work in [OAIS09] lies in how a fluid video is decomposed: we currently decompose a video into the average image and the differences between the average image and the original frames. On one hand, our prior work decomposed the differences further into the stationary motion field and the residuals. We reported that such a decomposition might cause a failure of transferring the residuals, in applying them to the fluid in the input image. More specifically, if the original video example was not appropriately chosen by the user, our prior approach would provide animations with visible artifacts, such as more noises and greater viscosity than the original video example (see Section 5.2 and 6 in [OAIS09]). Alternatively our new method simplifies the decomposition process, while having a wider variety of the fluid animations. These advantages are obtained by the new algorithms with the fluid video database, not with a single video specified by the user.

The supplementary video shows a side-by-side comparison of Figure 11-a using the previous and the present method. Previously, we could not reproduce fluid features of the video example, e.g., dynamic water splashes: the results appeared as if irrelevant synthesized noise flowed along a static motion field. Note that the motion of the resulting animation is faster than the motion of the video example; the residuals over a slow motion field were transferred to a fast motion field, which failed to reproduce the fluid features of the video example. To synthesize better fluid features, the user could search for a better video example, but it was time-consuming to search for a video with motion that was actually similar to that designed by the user. On the other hand, the present method successfully reproduced fluid features of the assigned video example pieces.

The present method further reduces the user's burden, as seen in the case of Figure 1. Here, using the previous method, the image had to be divided into waterfall and river parts and various videos had to be assigned independently; in contrast, the present method processes the whole image at once.

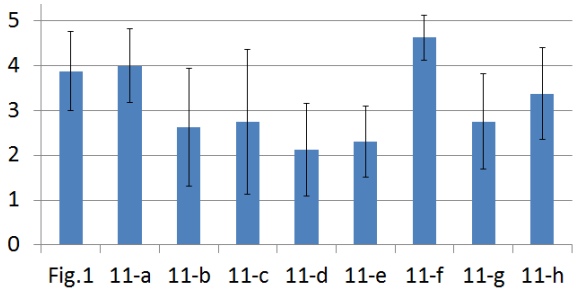


Figure 12: *The result of the user study. Each blue bar represents the average score of visual quality. Each error bar represents the standard deviation.*

7.2. User Study and Quality Discussion

We performed a user study to evaluate the visual quality of the resulting animations. Two professional animators and fourteen computer science students participated in the study. We presented the resulting animations of Figures 1 and 11. Each subject ranked the visual quality of each animation with regard to visible artifacts on a scale ranging from 1 (poorest quality) to 5 (best quality). The ranking is relative, i.e., 1 and 5 must each be assigned to at least one of the resulting animations. The results are shown in Figure 12. The highest average score with the smallest standard deviation was assigned to Figure 11-f, i.e., almost all subjects agreed that the water scene had a natural appearance. The other water scenes received no scores of 1 but several scores of 5, confirming acceptable visual quality. On the other hand, fire and smoke scenes were given lower scores. We discuss the

reasons and the limitations of our method, considering Figure 11-d as a typical case in which our method failed.

Failure of Video Example Assignment Figure 11-d had the lowest average score. The subjects commented that it looked noisy: the smoke differed in appearance between areas and the smoke movements were chaotic. This was due to failure of the assignment process for two major reasons. First, the selection of 100 candidates of video example pieces was not successful, as the target image was non-photorealistic and had many flat areas. Second, the assigned video example pieces had relatively chaotic movements. We specified the orientation and speed maps, which constrained the average motion of each patch and created the globally upward motions of the smoke. However, these specifications did not constrain the local motions, which were almost random. Thus, we require more sophisticated control of fluid motion than a single manually specified motion field.

Visible Artifacts Another limitation of the present method is that the resulting animation inherits artifacts of the video examples in the database. In our experiment, each video example was originally compressed by the QuickTime movie codec. Our system extracts each frame and saves it in JPEG format for efficient image processing. Therefore, block noise and interlacing artifacts are included in the database. These artifacts are not only inherited but are also sometimes exaggerated when appearance transfer process increases the image contrast. For example, the subjects commented that the bottom part of Figure 11-d had a noisy appearance. Some exaggerated JPEG block noise can actually be seen in this image (Fig. 13). We are currently considering simple solutions to prepare large storage and apply lossless compression to each video example.

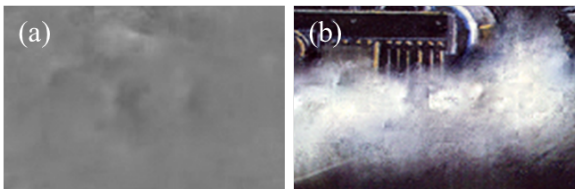


Figure 13: *(a) The assigned video example pieces had relatively low contrast. (b) Appearance transfer enhanced the contrast to recover the appearance of the target image, but it also exaggerated block noises that were hidden in (a).*

Preservation of Original Appearance Some subjects commented that the resulting animation looked different from the fluid in the targedt image. It is often difficult to preserve the appearance of the target image, especially when it has strong edges at significantly high frequencies. For example, Figure 11-c shows the characteristic appearance of smoke trajectories. However, its resulting animation does not retain this appearance, i.e., the trajectories disappeared. The same discussion is possible for the fire animation of Figure 11-e in that the original shape of the flame was not reproduced exactly. Our method is limited in that it preserves the overall shapes of fire and smoke, but our database is not sufficiently large to preserve details, such as the smoke trajectories.

Lighting Effect The lighting effect or the movement of an object other than fluid is outside our scope of this study. It is often noted that animations in which only fluid parts are mobile and other parts remain completely stationary do not have a natural appearance. Especially, the movement of the flame in Figure 11-e must influence the lighting effect on the ground. However, our goal is not to animate the whole scene of a target image but to animate fluid parts.

8. Conclusions and Future Work

We have developed a data-driven method for synthesizing a fluid animation from a single image. User input includes a target image and an alpha matte that extracts the fluid region of interest. A few additional suggestions regarding fluid speed or orientation enable the user to control and refine the resulting animation. The technique described here allows the user to produce a higher quality fluid animation with less effort than with other methods reported previously. The present approach suggests a number of areas for future work. In particular, we are currently pursuing three avenues as described below:

More Efficient Use of Video Database One of the limitations of our method is the difficulty of preserving detailed appearance of a target image. This is because our database does not include a video example that fits a target appearance. In addition to increasing the size of the database, we plan to investigate more flexible use of the video database. For example, the patch size is currently fixed to 48×48. However, if we assigned a video example piece taking its image warping into account as described previously [BSHK04], more detailed target appearance would be reproduced.

A More Intelligent System Our current methodology involves cutting a video example into pieces and assigning them locally. However, most failed assignments were caused by the loss of global information. If the system could understand scene structures or fluid states, our approach would become more automated. That is, the system would be able to estimate fluid velocity by understanding the scene structure, which would allow it to distinguish, for example, chimney smoke from the smoke caused by an explosion, even though they look similar. Previous studies have attempted to develop more intelligent systems by incorporating SIFT flow [LYT*08]. This type of system can estimate the motion of a campfire in a single image by fitting of a video sample. We plan to develop a similar technique and incorporate it into our system.

More Dynamic Fluid Animation The present method works well in treating the high-frequency part of the fluid, such as water splashes. However, large fluid features, such as the overall shape of a breaking wave, cannot be synthesized. Such global features of the fluid should be created in the future. We plan to investigate image object-retrieval techniques as described previously [KCCH09] to efficiently extract large semantic regions and assign it to parts of the target image.

References

[BA83] BURT P. J., ADELSON E. H.: A multiresolution spline with application to image mosaics. *ACM Trans. Graph. 2*, 4 (1983), 217–236. 6

[BE01] BROSTOW G. J., ESSA I.: Image-based motion blur for stop motion animation. In *Proc. SIGGRAPH 2001* (2001), pp. 561–566. 6

[BJEYLW01] BAR-JOSEPH Z., EL-YANIV R., LISCHINSKI D., WERMAN M.: Texture mixing and texture movie synthesis using statistical learning. *IEEE Trans. Vis. Comput. Graph. 7*, 2 (2001), 120–135. 2

[BNTS07] BOUSSEAU A., NEYRET F., THOLLOT J., SALESIN D.: Video watercolorization using bidirectional texture advection. In *Proc. SIGGRAPH 2007* (2007), p. 104. 7

[BSHK04] BHAT K. S., SEITZ S. M., HODGINS J. K., KHOSLA P. K.: Flow-based video synthesis and editing. *ACM Trans. Graph. 23*, 3 (2004), 360–363. 2, 8, 9

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (2001), 1222–1239. 6

[CCT*09] CHEN T., CHENG M.-M., TAN P., SHAMIR A., HU S.-M.: Sketch2photo: internet image montage. In *Proc. SIGGRAPH Asia 2009* (2009), pp. 1–10. 2

[CGZ*05] CHUANG Y.-Y., GOLDMAN D. B., ZHENG K. C., CURLESS B., SALESIN D. H., SZELISKI R.: Animating pictures with stochastic motion textures. In *Proc. SIGGRAPH 2005* (2005), pp. 853–860. 1, 2

[DCWS03] DORETTO G., CHIUSO A., WU Y. N., SOATTO S.: Dynamic textures. *Int. J. Comput. Vision 51*, 2 (2003), 91–109. 2, 8

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. pp. 341–346. 3

[HAA97] HORRY Y., ANJYO K., ARAI K.: Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proc. SIGGRAPH '97* (1997), pp. 225–232. 1, 2, 7

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *SIGGRAPH1995* (1995), pp. 229–238. 7

[HDK07] HORNUNG A., DEKKERS E., KOBBELT L.: Character animation from 2d pictures and 3d motion data. *ACM Trans. Graph. 26*, 1 (2007), 1. 1, 2

[HE07] HAYS J., EFROS A. A.: Scene completion using millions of photographs. In *Proc. SIGGRAPH 2007* (2007), p. 4. 2, 3

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *Proc. SIGGRAPH 2001* (2001), pp. 327–340. 3

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (2005), 1134–1141. 1, 2

[KCCH09] KUO Y.-H., CHEN K.-T., CHIANG C.-H., HSU W. H.: Query expansion for hash-based image object retrieval. In *ACM Multimedia* (2009), pp. 65–74. 10

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. In *Proc. SIGGRAPH 2005* (2005), pp. 795–802. 2, 8

[KSH04] KE Y., SUKTHANKAR R., HUSTON L.: Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia* (2004), pp. 869–876. 8

[LEN09] LALONDE J.-F., EFROS A. A., NARASIMHAN S. G.: Webcam clip art: appearance and illuminant transfer from time-lapse sequences. *ACM Trans. Graph. 28*, 5 (2009), 1–10. 2

[LP05] LI F.-F., PERONA P.: A bayesian hierarchical model for learning natural scene categories. In *Proc. of CVPR '05* (2005), pp. 524–531. 4

[LSP06] LAZEBNIK S., SCHMID C., PONCE J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. of CVPR '06* (2006), pp. 2169–2178. 5

[LSTS04] LI Y., SUN J., TANG C.-K., SHUM H.-Y.: Lazy snapping. *ACM Trans. Graph. 23*, 3 (2004), 303–308. 7

[LWW*07] LIN Z., WANG L., WANG Y., KANG S. B., FANG T.: High resolution animated scenes from stills. *IEEE Trans. Vis. Comput. Graph. 13*, 3 (2007), 562–568. 2

[LYT*08] LIU C., YUEN J., TORRALBA A., SIVIC J., FREEMAN W. T.: Sift flow: Dense correspondence across different scenes. In *ECCV 2008* (2008), pp. 28–42. 2, 9

[MPK09] MOHAMMED U., PRINCE S. J. D., KAUTZ J.: Visiolization: generating novel facial images. *ACM Trans. Graph. 28* (2009). 4

[MWGZ09] MA C., WEI L.-Y., GUO B., ZHOU K.: Motion field texture synthesis. *ACM Trans. Graph. 28*, 5 (2009), 1–8. 2

[OAIS09] OKABE M., ANJYO K., IGARASHI T., SEIDEL H.-P.: Animating pictures of fluid using video examples. *Computer Graphics Forum (Proc. EUROGRAPHICS) 28*, 2 (2009), 677–686. 1, 2, 6, 7, 8

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proc. SIGGRAPH 2000* (2000), pp. 489–498. 2, 8

[SSSFF09] SUN M., SU H., SAVARESE S., FEI-FEI L.: A multi-view probabilistic model for 3d object classes. pp. 1247 –1254. 4

[SZS*08] SZELISKI R., ZABIH R., SCHARSTEIN D., VEKSLER O., KOLMOGOROV V., AGARWALA A., TAPPEN M., ROTHER C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 6 (2008), 1068–1080. 6

[TYS09] TAO L., YUAN L., SUN J.: Skyfinder: attribute-based sky image search. In *Proc. SIGGRAPH 2009* (2009), pp. 1–5. 2, 3

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proc. SIGGRAPH 2000* (2000), pp. 479–488. 2

[WZ03] WANG Y., ZHU S.-C.: Modeling textured motion: Particle, wave and sketch. In *ICCV2003* (2003), pp. 213–220. 2

[ZKF05] ZHAO Y., KARYPIS G., FAYYAD U.: Hierarchical clustering algorithms for document datasets. *Data Min. Knowl. Discov. 10*, 2 (2005), 141–168. 4

[ZPB07] ZACH C., POCK T., BISCHOF H.: A duality based approach for realtime tv-l1 optical flow. In *Pattern Recognition (Proc. DAGM)* (Heidelberg, Germany, 2007), pp. 214–223. 4