EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Animating Pictures of Fluid using Video Examples

Makoto Okabe[1] Ken Anjyo[2] Takeo Igarashi[3] Hans-Peter Seidel[1]

[1]Max Planck Institut für Informatik [2]OLM Digital, Inc. [3]The University of Tokyo, JST/ERATO

Abstract

We propose a system that allows the user to design a continuous flow animation starting from a still fluid image. The basic idea is to apply the fluid motion extracted from a video example to the target image. The system first decomposes the video example into three components, an average image, a flow field and residuals. The user then specifies equivalent information over the target image. The user manually paints the rough flow field, and the system automatically refines it using the estimated gradients of the target image. The user semi-automatically transfers the residuals onto the target image. The system then approximates the average image and synthesizes an animation on the target image by adding the transferred residuals and warping them according to the user-specified flow field. Finally, the system adjusts the appearance of the resulting animation by applying histogram matching. We designed animations of various pictures, such as rivers, waterfalls, fires, and smoke.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Methodology and Techniques—Interaction techniques—; Image Processing and Computer Vision [I.4.8]: Scene Analysis—Motion—;

1. Introduction

Contemplating a photo or painting of natural scene, we often imagine how the scene in the image would appear animated. Several papers discussed how to animate such still images [FAH91, LW94, BC02, CGZ*05, IMH05], but animating fluids in a still image remains a great challenge. This is also one of the practical demands commonly desired in the digital image industries. For example, in a cartoon animation or even in a live action film, fluid scene images are often used as the background of moving characters. The background should then be animated as well, to make the whole scene more dramatic or impressive, but this may not be feasible if the creation of the animated background is demanding.

There are two major methods for creating a flow animation: one is a fluid simulator, and the other is editing existing videos. Fluid simulators allow users to create a wide variety of fluids [FM96, FF01]. Several methods also allow users to specify keyframes [TMPS03, MTPS04]. However, since these methods are computationally expensive and have many parameters to be set appropriately, even experienced designers must spend considerable time by trial and error to achieve satisfactory results. The other method is to copy and paste existing video fragments into the target image. However, available videos do not always match the appearance of the target image. Video texture synthesis methods are another possibility [SSSE00, WL00, BJEYLW01, DCWS03, WZ03, BSHK04, KEBK05, LWW*07, NKL*07], but no existing method has successfully synthesized a continuous flow animation in a given still fluid image while preserving the overall appearance.

We propose a system that allows the user to design a continuous flow animation starting from a fluid picture (Fig. 1). Given a target image, the user gives the system three inputs. First, the user selects a video example that specifies the desired fluid motion. Second, the user manually paints a rough flow field that is automatically refined using the estimated gradients of the target image. Last, the user semi-automatically transfers high-frequency fluid features from the video example to the target image. With our system and a library of video examples containing typical fluid sequences, one can quickly add reasonable motion to a fluid in an image without using a complicated fluid simulation.

Our technical contribution is in the way we apply the fluid motion from a video example to the target image. Continuous flow animation has roughly stationary temporal dynamics [DCWS03, BSHK04], and we assume that the velocity at a single fixed point of our target animations is constant over time. Based on this assumption, our system first decomposes

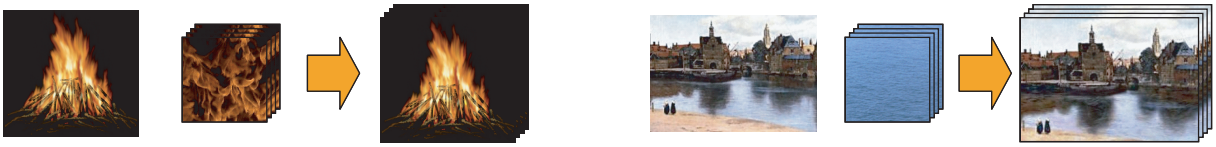


Figure 1: *Our system allows the user to design a continuous flow animation starting from a fluid picture by semi-automatically applying the fluid motion extracted from the video example. The left example animates a campfire in our artist's painting using a fire video. The right example animates a water surface in Johannes Vermeer's painting using an ocean video.*

the video example into three components, an average image, a flow field, and residuals. The flow field is obtained as an average of optical flows computed by comparing neighboring frame pairs. The residuals are obtained as differential images between the original frames and corresponding warped frames, representing non-translational high-frequency information. After refining the flow field and transferring the residuals onto the target image, the system then approximates the average image and synthesizes an animation by adding the transferred residuals and warping them according to the designed flow field. Finally, the system applies histogram matching to preserve the original appearance of the target image.

Our main contribution is the overall design of the method based on three assumptions: first, a human can easily imagine a rough motion in a fluid picture, second, a human can also easily find a video example with a similar motion to the fluid picture, and last, the computer can automatically integrate the user-specified rough hints to synthesize a flow animation in the fluid picture. Based on these notions, we provide user interfaces to efficiently specify decomposed information of continuous flow animation over the target image. Our approach allows an appearance-based design process and makes the authoring of flow animation accessible to non-professional users. Our system can also be a powerful authoring tool for a professional designer who wants to specify every detail of the motion.

2. Previous Work

Many methods have been proposed for the creation of animations from a single image. Freeman *et al.* proposed "Motion without Movement," where the user can create the illusion of motion in a single image [FAH91]. Litwinowicz and Williams's method allows the user to create two-dimensional (2D) animations using line drawings for image deformation as keyframes [LW94]. Barrett and Cheney proposed an integrated system to edit 2D single images, which enables the user to create 2D animations [BC02]. Igarashi *et al.* proposed an interactive image manipulation method that enables the user to create 2D animations interactively by operating user-specified handles [IMH05]. These methods are useful for creating 2D character animations but 2D fluid animations are out of their focuses. Chuang *et al.* proposed a method to synthesize a video with stochastic motions from a single image [CGZ*05]. This method addresses the problem of animating single images representing natural phenomena but supports not flowing fluid animation but only oscillation of water surface like ripple.

Video texture synthesis has also been well-explored. Wei and Levoy proposed a fast non-parametric algorithm to synthesize three-dimensional (3D) spatial-temporal volumes [WL00]. Bar-Joseph *et al.* proposed a stochastic algorithm to synthesize a video sequence using multi-resolution analysis [BJEYLW01]. To create long videos from short clips, the video texture method concatenates appropriately chosen subsequences [SSSE00], and the dynamic texture method uses autoregressive filters [DCWS03]. These methods enable the user to extend an existing video spatially and temporally. However, it is difficult for the user to modify the appearance or motion of the synthesized video. Wang and Zhu analyzed fluid animation and represented it with textons to synthesize video sequences [WZ03]. Bhat *et al.* proposed a sketching interface and an algorithm to synthesize a fluid animation from a video example [BSHK04], where the user can also change the appearance of the fluid. However, the problem of animating a fluid picture has not been addressed. Kwatra *et al.* proposed a method to design a texture animation flowing over a user-specified flow field [KEBK05]. Narain *et al.* transferred video textures to 3D fluid surfaces or other video sequences [NKL*07]. These methods enable the user to modify the existing video and create a novel fluid animation, although they do not allow the user to specify an appearance constraint using a single image. Hashimoto *et al.* extended the image analogies technique [HJO*01] for example-based video filtering [HJN03]. This method addresses the problem of designing the appearance of a video sequence using example still images. In contrast, our goal is to design the motion of still images using video examples. Lin *et al.* proposed a method to produce an animation from not single but multiple high resolution stills [LWW*07].

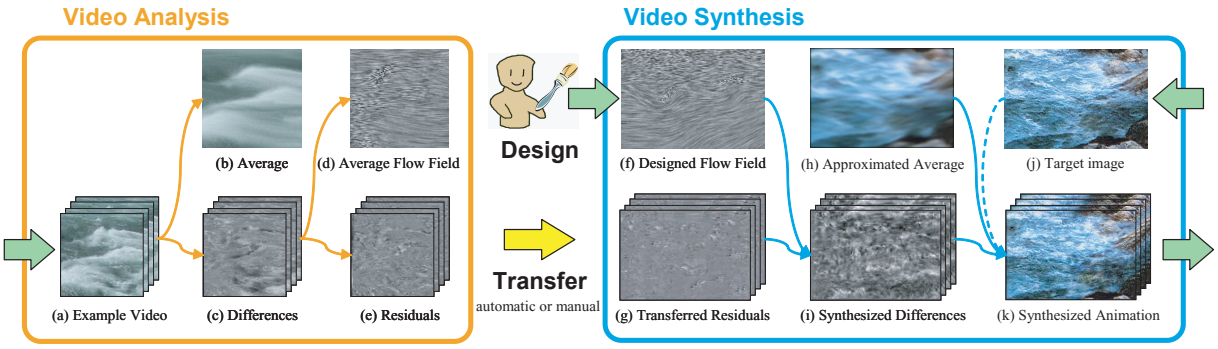


Figure 2: *System overview. Green arrows represent input and output of our system. The video example (a) is decomposed into the average image (b) and the differences between the original frames and the average image (c). The differences are then decomposed into the flow field (d) and the residuals (e). The user semi-automatically designs the flow field (f) and transfers the residuals (g). The system then computes the approximated average image (h) by applying motion blur to the target image, an cutout from Thomas Moran's painting (j). The synthesized differences (i) are computed by combining the designed flow field and transferred residuals. Finally, to preserve the original appearance of the target image (j), the system applies histogram matching (dashed line) to compose the final flow animation (k).*

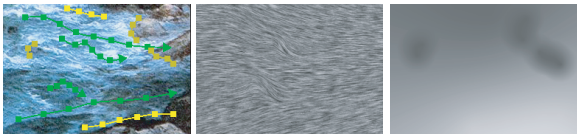
3. System Overview

We give an overview of the process of designing a flow animation. We then describe the two important processes of our approach, flow field design and transfer of residuals.

Fig. 2 presents the system overview. Given the target image (Fig. 2-j), a video example whose fluid motion the user wants to apply to the target image must be selected (Fig. 2-a)). The analysis of the video example is performed (Section 4) . The video example is first decomposed into the average image through all the frames (Fig. 2-b) and the differences between the original frames and the average image (Fig. 2-c). The differences are then decomposed into the flow field (Fig. 2-d) and the residuals (Fig. 2-e). Since we assume our target animations have stationary temporal dynamics, the flow field is obtained as an average of optical flows computed by comparing neighboring frame pairs. The residuals are obtained as differential images between the original frames and corresponding warped frames according to the flow field. Given the decomposed information, the user then semi-automatically specifies equivalent information over the target image (Section 5). The user manually designs the rough flow field in the target image by painting strokes, which is then refined automatically using the estimated gradients of the target image (Fig. 2-f, Section 3.1). Our system automatically transfers the residuals according to an assumption that small patches having similar flow fields would have similar residuals, and the user can also transfer the residuals manually by painting operations (Fig. 2-g, Section 3.2). The approximated average image is then obtained by applying motion-blur to the target image using the designed flow field (Fig. 2-h). The system adds the transferred residuals and warps them according to the designed flow field to synthesize the differences (Fig. 2-i). The system then combines the approximated average image and the synthesized differences to give an animation. Finally, to preserve the original appearance of the target image (Fig. 2-j), the system applies histogram matching to the animation to synthesize the resulting flow animation (Fig. 2-k). When the target image has multiple fluids and other still objects, an alpha matte is required to extract fluid parts from the scene.

The decomposition of flow animation into three design factors is our key idea that allows the user an efficient and intuitive design process. First, the extraction of the average image enables to easily replace the overall appearance of a flow animation. The flow field is an important characteristic of a flow animation and its extraction as a design factor is essential. Finally, the extraction of residuals allows the user to transfer desired small fluid features of a video example efficiently. The residuals contain only the non-translational high-frequency information that the flow field cannot capture, such as water drops that suddenly appear or disappear. An expert user of a video authoring tool may copy small patches of a video example without any decomposition, paste them onto the target image, and adjust their appearances. However, adjustment of appearance is a difficult task. Moreover, such naive cut and paste cause visible discontinuities between patches, because each patch has independent fluid features. On the other hand, since our decomposition allows such naive cut and paste on the residuals, we provide intuitive tools to transfer the residuals (Section 3.2).

Bhat *et al.* decomposed a flow animation into two components, flow field along user-specified strokes and particles of video textures, which allow to edit an existing video efficiently . The particles possess small fluid features and play a similar role to our residuals. The difference is the particles move along the flow field but our residuals are stationary.



(a) User Input (b) Orientation (c) Speed

Figure 3: *Designing a rough flow field by painting. (a) Strokes of Orientation Brush and Speed Brush are green and yellow respectively. Dark strokes are painted in local mode. (b) The designed orientation component is visualized using Line Integral Convolution. (c) The designed speed component is visualized with faster motion having brighter colors.*

3.1. Flow Field Design

Flow field design consists of two processes: manual design of rough flow field and its automatic refinement. The manual design process allows the user to paint the flow field roughly. The automatic process refines the flow field using the estimated gradients of the target image. Since image gradients don't always hold correct information of our target flow field, we propose the semi-automatic method that allows the user to design a complex flow field efficiently.

To make the manual design process intuitive, we decompose the flow field into orientation and speed components and provide a separate brush for each of them (*Orientation Brush* and *Speed Brush*). Orientation Brush assigns an orientation vector to each pixel under the brush stroke. The orientation vector is tangential to the stroke. To intuitively specify a speed value with Speed Brush, the user can fetch the speed value from the video example. The sparsely painted orientations and speeds (shown as bright green and yellow strokes in Fig. 3-a) are interpolated over the target image. The user can also edit the flow field locally. A brush stroke painted in local mode (shown as dark green and yellow strokes in Fig. 3-a) affects its nearby pixels [SWHS97] and its effect is controlled by brush size and alpha parameters. Fig. 3-b and c show each component of the designed flow field.

3.2. Transfer of Residuals

Automatic Transfer of Residuals. The system can automatically transfer residuals of the video example to the target image. We assume that when a small patch of the target image has a similar flow field to a patch of the video example, both patches have similar residuals. For each patch of the target image, we perform the best match search between the flow fields of the video example and the target image (Fig. 4). The system then copies fragments of residuals. The user specifies the patch size and runs several seconds of the process. The patch size is typically from 8×8 to 32×32 pixels. Automatic transfer is useful for inexperienced users who want to obtain quick results with minimum intervention.

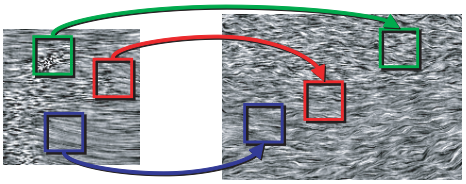


Figure 4: *Best match search between the flow fields of the video example (left) and the target image (right).*

Manual Transfer of Residuals. We provide a *Clone Brush* to manually transfer residuals. Using the Clone Brush, the user draws a *source stroke* on the video example and a *destination stroke* on the target image. The system cuts out the residuals around the source stroke and pastes them around the destination stroke. The system deforms the transferred region so that the source stroke fits to the destination stroke. Brush size and alpha parameters are used to control the effect of the Clone Brush on the underlying residuals. Manual transfer is useful for expert users who want to control details.

4. Video Analysis

In this section, we describe the algorithms to analyze a video example and decompose it into three components of an average image, a flow field and residuals (Fig. 5).

4.1. Average Image

We first average all the frames of the video example as $A = \frac{1}{N}\sum_{i=0}^{N-1} F_i$, where F_i are the frame of the video example and N is its number (Fig. 5-a and b). We then compute the differences as $D_i = F_i - A$ (Fig. 5-c).

4.2. Flow Field

The flow field is obtained as an average of optical flows computed by comparing neighboring frame pairs of the differences D. An optical flow is computed by using the Lucas-Kanade method with an image-pyramid representation [LK81, Bou99]. It takes frame D_i and the following frame D_{i+1} as input and generates a vector field. The method has two steps, feature extraction and optical flow computation. Each step has an open parameter, the number of features and the window size. We set the number of features to 100 and the window size to 10. The computed optical flow is sparse, and only the feature points have velocities (Fig. 5-d). To obtain a continuous optical flow, we interpolate them over the image space using radial basis function $\phi(x) = |x|$ [PFH00]. Finally, we average the optical flows through all the frames to obtain the flow field (Fig. 5-e).

4.3. Warping Function

We define the warping function W that takes an image I and warps it according to the flow field. The warped image is

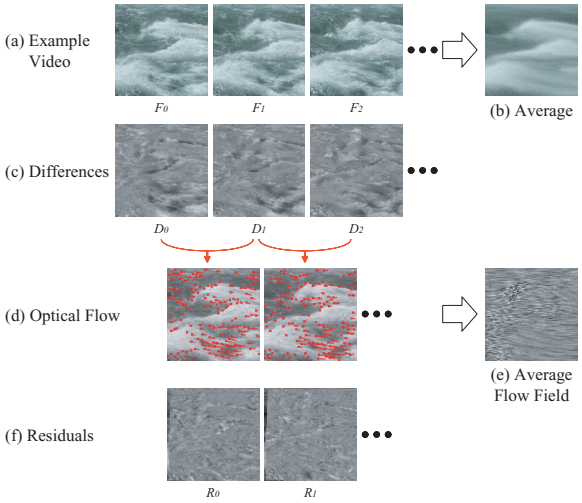


Figure 5: *Video analysis. For differences and residuals, we visualize each image so that gray color represents zero values; darker pixels have negative values, and brighter pixels have positive values.*

denoted as $W(I)$. Our warping operation is based on backward mapping. Let $\vec{x} = (x, y)$ be a position and $p(\vec{x}, i)$ be the pixel value at $\vec{x}$ in i-th frame. Given a flow field defined as $\vec{v}(\vec{x})$, each pixel value of the warped image is computed as $p(\vec{x}, i) = p(\vec{x} - \vec{v}(\vec{x}), i-1)$. In practice, this operation is achieved by creating a regular triangular mesh over the image space, deforming the mesh via $\vec{v}$, and sampling the pixel values of $(i-1)$-th frame via barycentric coordinates.

4.4. Residuals

To analyze non-translational high frequency information, we consider the differential image R_i between the warped i-th frame and the $(i+1)$-th frame (Fig. 5-f):

$$R_i = D_{i+1} - W(D_i). \tag{1}$$

4.5. Video Reconstruction

The flow field defines a velocity at each pixel, and the residuals contain only the non-translational high-frequency information that the flow field cannot capture, such as water drops that suddenly appear or disappear. Given the first frame D_0, the warping function W, and the residuals R_i, the differences D_i are reconstructed by rewriting Eq. 1 as $D_{i+1} = W(D_i) + R_i$. We introduce a composite function W^a of W and eliminate the recurrence.

$$W^0(I) = I, \tag{2}$$

$$W^{n+1}(I) = W \circ W^n(I) \qquad n \geq 0, \tag{3}$$

$$D_i = W^i(D_0) + \sum_{k=0}^{i-1} W^{i-k-1}(R_k). \tag{4}$$

Eq. 4 means that the first frame and each residual affect all the following frames. When i is large, we can ignore the effect of the first frame and very old residuals. We can limit the life time of each residual with an open parameter τ:

$$D_i \approx \sum_{k=i-\tau}^{i-1} W^{i-k-1}(R_k) \qquad \tau \leq i < N. \tag{5}$$

Finally, the video example is reconstructed as follows:

$$F_i \approx A + \sum_{k=i-\tau}^{i-1} W^{i-k-1}(R_k) \qquad \tau \leq i < N. \tag{6}$$

Eq. 6 means that we can synthesize a flow animation of the target image in the same manner, by appropriately specifying an average image, a warping function and residuals over the target image. Since we have denoted the decomposed information as A, W and R, we denote the specified information as A', W' and R'.

5. Video Synthesis

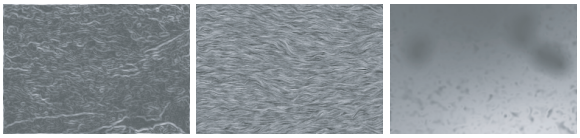
In this section, we describe the algorithms for flow field design specifying W', the automatic and manual transfer of residuals specifying R', the approximation of the average image specifying A', and the video reconstruction and histogram matching synthesizing the resulting flow animation.

5.1. Flow Field Design and Warping Function

As for manual design of rough flow field, Orientation Brush and Speed Brush specify orientation vectors and speed values at pixels under each stroke. Given sparsely specified values, we interpolate them over the target image space using the radial basis function $\phi(x) = |x|$ [PFH00]. For the orientation vector, we interpolate x- and y-components separately and normalize the vector at each pixel. In local mode, we make a layer of interpolated values and an alpha matte of user-specified brush size and alpha parameters for each brush. We then integrate all the layers with alpha mattes to give the rough flow field. We denote the rough orientation and speed components as Φ_o and Φ_s (Fig. 3-b and c).

The automatic refinement of the flow field is inspired by image-based non-photorealistic rendering method [HE04] that estimates brush stroke orientations of painting based on gradients in an input image. The automatic refinement process begins by calculating the gradients of the target image. The target image is blurred by a Gaussian kernel to remove the noise, and the gradients G are then estimated with a Sobel filter [FP02] (Fig. 6-a). We set the kernel size to 5×5 in our experiments.

We compute the refined orientation component as Ψ_o so that $\Psi_o(\vec{x})$ is a unit vector parallel to $G(\vec{x})$ and the dot product $\Psi_o(\vec{x}) \cdot \Phi_o(\vec{x})$ is larger than zero. Since Ψ_o is noisy as an orientation field, we blur Ψ_o by the other Gaussian kernel of width w_f and obtain a smooth orientation component (Fig. 6-b). w_f is an open parameter that controls



(a) (b) (c)

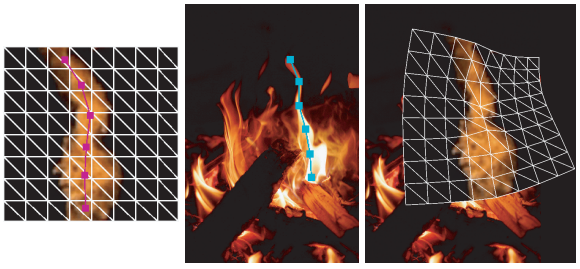
Figure 6: *Refinement of the flow field. (a) The estimated gradients of the target image (Fig. 3-a). (b) Refined orientation component. (c) Refined speed component.*

the sensitivity of the refined flow field to the image gradients. We then compute the refined speed component as $\Psi_s(\vec{x}) = \Psi_o(\vec{x}) \cdot \Phi_o(\vec{x}) \cdot \Phi_s(\vec{x})$ (Fig. 6-c). Our refined orientation component is strongly dependent on G, and Ψ_o is just used to determine its alternative directions. Our refined speed component is determined so that gradients parallel to the user-specified flow field keep their speeds, but gradients orthogonal to it degrade their speeds. Finally, the corresponding warping function W' is defined based on the refined flow field $\Psi_o \cdot \Psi_s$.

5.2. Automatic Transfer of Residuals

We perform automatic transfer with respect to each rectangular patch in the flow field over the target image by finding the best matching exemplar window in the flow field over the video example (Fig. 4). The same matches are used for transferring all the frames of residuals to maintain temporal coherency. The best match search is a standard nearest-neighbor search in a high-dimensional space. We accelerate this step by using principle component analysis (PCA) and approximated nearest neighbor method (ANN) [HJO*01]. Further acceleration is possible using down-sampled target images and video examples. In our experiments, we reduced the image resolution by two- or four-fold downsampling. Since the patches on the target image overlap, we fill the image space with representative patches in a Lapped Texture manner [PFH00]. This method successfully preserves the high-frequency features of transferred residuals. We first sort all the patches according to their similarities; our similarity measure is an L2-norm between the patch in the target image $\vec{s}$ and the patch in the video example $\vec{e}$ given as $|\vec{s} - \vec{e}|^2$. We then paste the patches in the sorted order. The boundary of each patch is alpha masked and blended.

The automatic transfer is inspired by the idea of Image Analogies [HJO*01]. We assume that the transferred residuals relate to the designed flow field in the same way as the original residuals relate to the average flow field of the video example. This assumption is introduced to preserve the fluid features of the original video example in the synthesized differences. Otherwise, the residuals transferred randomly without considering the relationship tend to cause random noises in the synthesized differences.



(a) (b) (c)

Figure 7: *Manual transfer of residuals using shape deformation. (a) A triangular mesh and the source stroke on the video example. (b) The destination stroke on the target image. (c) The deformed mesh over the target image.*

5.3. Manual Transfer of Residuals

Manual transfer copies residuals from the region around a source stroke to the region around a destination stroke. To deform the region, we first create a triangular mesh over the video example (Fig. 7-a). We then apply as-rigid-as-possible shape deformation using the vertices of the source and destination strokes as positional constraints [IMH05]. We transfer residuals by sampling pixel values in the deformed mesh via barycentric coordinates. The same mesh is used for transferring all the frames of residuals to maintain temporal coherency. We make an alpha matte of user-specified brush size and alpha parameters to blend the transferred residuals with underlying residuals.

5.4. Approximation of the Average Image

We first approximate the average image of the target flow animation by applying image-based motion blur [BE01] and Gaussian blur to the target image. Fig. 8-a shows the target image, and Fig. 8-b shows the target image motion-blurred using the designed flow field. An open parameter to simulate the shutter speed controls the effect of motion blur. Since motion blur makes sharp edges along the flow field and they will cause visible artifacts in the resulting flow animation, we apply additional Gaussian blur and obtain the smoother image (Fig. 8-c). The width of Gaussian kernel w_a is an open parameter. We use the resulting image as the approximated average image A'.

5.5. Video Reconstruction

Given the warping function of the designed flow field W', and the transferred residuals R'_i, we can reconstruct the differences D'_i by introducing a composite function of W' as W'^a and the life time parameter τ:

$$D'_i = \sum_{k=i-\tau}^{i-1} W'^{i-k-1}(R'_k). \quad (7)$$

(a) (b) (c)

Figure 8: *Approximation of the average image. (a) The target image. (b) The motion-blurred image with a shutter speed set to 0.1 millisecond. (c) The approximated average image obtained by applying Gaussian-blur.*

We then adjust the contrast and brightness of D_i' as

$$D_i'' = (D_i' - \langle D' \rangle) \cdot \alpha + \beta, \tag{8}$$

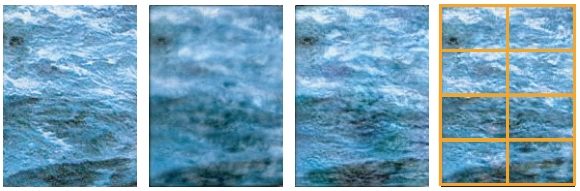
where $\langle D' \rangle$ is the mean of all the pixel values of D', α and β are the contrast and brightness parameters. Let C the difference between the target image and the approximated average image A', since the differences D_i'' should have the same contrast and brightness as C, the brightness β is computed as the mean of all the pixel values of C. The contrast α is computed using the standard deviations of C and D' as $\alpha = \sigma_C / \sigma_{D'}$.

We then combine the approximated average image A' and the synthesized differences D_i'' to give a flow animation as $F_i' = A' + D_i''$. When the video example has N frames, the duration of the reconstructed animation is $N - \tau$ frames, since both the residuals R and the transferred residuals R' have $N - 1$ frames and F_i' is computed from F_τ' to F_{N-1}'.

5.6. Histogram Matching

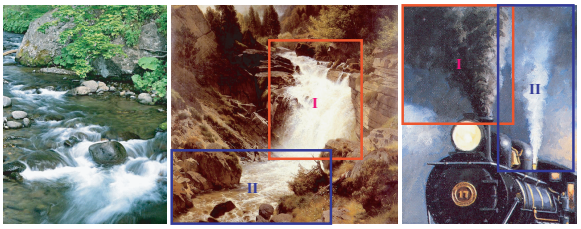
The reconstructed animation usually has blurred frames or different appearance from the target image. We apply histogram matching to recover reduced contrast and original appearance of each frame. We use Heeger and Bergen's texture synthesis algorithm [HB95]. With this algorithm, we construct steerable pyramids of both the target image and each reconstructed frame. We then apply histogram matching to histograms of steerable coefficients in each sub-band. This process is repeated several times. Fig. 9-c shows the result of this process. When we must preserve the original appearance more locally, we divide the image space into grids and apply the process to each grid one by one, and combine all the grids in the image space again. To remove visible discontinuities between neighboring grids, we apply alpha blending to their overlapping boundaries. Fig. 9-d shows the result.

We try to exploit the high frequency information of the target image as much as possible by applying histogram matching with grids. Because our approach uses a video example, there is always a conflict between the high frequency information from the target image and the video example. Our method is designed so that both components are mixed in a balanced manner, by allowing the user to control the number of histogram matching and the size of grids.



(a) (b) (c) (d)

Figure 9: *Histogram matching to preserve the original appearance of the target image. (a) The target image. (b) The reconstructed frame. (c) The reduced contrast of the reconstructed frame is recovered by histogram matching. (d) The process is applied to each region in the orange grids (2×4) to preserve the original appearance more locally.*



Brook Waterfall Smoke

Figure 10: *Resulting flow animations.*

6. Results and Discussion

We applied our method to various types of fluid pictures. Fig. 1 and Fig. 10 show several examples of the resulting flow animations. Table 1 summarizes information related to the results. The accompanying video shows long, natural, continuous flow animations without any noticeable short loops or spatial discontinuities which are inevitable in the naive cut and paste approach.

Given a target image and its alpha matte, we launch the system and load them and the video example. We then start to paint the rough flow field. As soon as each painting operation is performed, the system automatically refines the flow field. We then transfer the residuals to the target image, and synthesize the first short animation. Since histogram matching with a steerable pyramid takes a long time, we adopt the Laplacian pyramid that renders an animation favorable as a preview. We apply histogram matching once for rendering the preview. We usually spend several minutes to create this first result (note that the time for alpha matte creation is ignored). If the result is unsatisfactory, we can go back to the design process and fine-tune the result. "Water Surface" in Johannes Vermeer's painting (Fig. 1) is one of the simplest examples, because its flow field does not have sig-

nificant speeds. By preparing the video example of a real ocean, we could design the animation by specifying a simple flow field and transferring the residuals automatically. For Johann Gottfried Steffan's waterfall painting ("Waterfall" in Fig. 10) and W W Stewart's steam train painting ("Smoke" in Fig. 10), we extracted each part from the target image and designed the flow animation using a corresponding video example respectively.

Making an Infinitely Flowing Animation. The duration of the synthesized flow animation is normally dependent on the duration of the video example. However, if the user can prepare an infinitely flowing animation as a video example [SSSE00, DCWS03, BSHK04], since the decomposed residuals do not repeat either, the resulting flow animation flows infinitely. For the river image shown in Fig. 2-j, we applied the video example of the other river that is synthesized by Bhat *et al.* without a loop (courtesy Bhat *et al.*).

Flow Field Design. The river and waterfall animations usually have complex flow fields. Our painting interface and automatic refinement are especially useful for designing such complex flow fields. As shown in Table 1, the numbers of used Orientation and Speed Brushes are small compared with the complexity of the target flow field. This is because the automatic refinement process helps the user to design the details of the target flow field efficiently. To create comparable results without refinement, the user has to paint detailed flow fields carefully using many brushes in local mode.

Transfer of Residuals. As shown in Table 1, since the automatic transfer based on the assumption of flow field similarity works well for most cases, we did not use many Clone Brushes. However, there are two cases where the manual transfer is useful. One is the case where the designed flow field is uniform and the automatic transfer synthesizes not variational but flat residuals. The other is the case where the video example has large features and the user wants to transfer them. For "Smoke" example, we used Clone Brushes to transfer large features of the video example.

Boundary Treatment. For the smoke and campfire animations, we paid much attention to boundary treatment. Smoke and fire usually change their shape dramatically. However, it is difficult to design a flow animation that has dynamic shape boundaries using our system, because the overall shape of the resulting animation is specified by the static approximated average image. Therefore, we need to carefully approximate the average image choosing appropriate parameters for motion-blur and Gaussian-blur. The alpha matte for extracting fluid parts from the target image must also be carefully designed. This is one of the limitations of our current method.

Video Example as Design Factor. One of the interesting aspects of our method is the video example being a design factor. We can design different flow animations from the same target image by applying different video examples. Fig. 11 shows that frames synthesized using different video examples have different appearances. For these frames, we used 3×3 grids and applied histogram matching only once to preserve the original fluid features of the video examples.

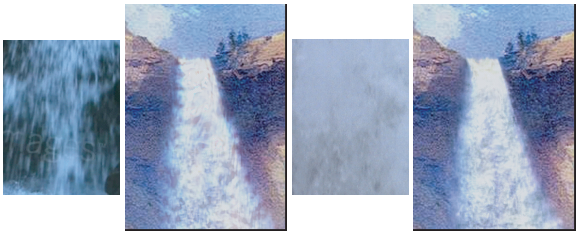


Figure 11: *Flow animation variations from the same target image. For the different video examples (a) and (c), the corresponding synthesized animations (b) and (d) are created by our method.*

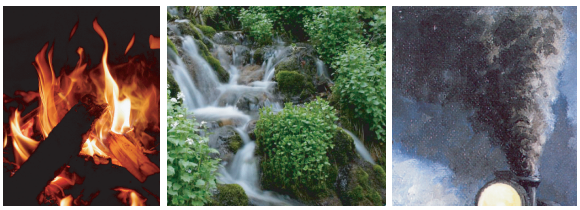
User Experience. We performed an informal user study to investigate the usability of our system. The subjects were five researchers in the computer science department who were novice users of our system. We asked them to design flow animations by using Orientation Brushes, Speed Brushes and adjusting the parameters of shutter speed and w_a for approximating the average image. The other parameters are set to their default values: w_f is 12, τ is 16, *HM Grid* is proportional to the resolution, and *HM Rep* is 3. The target image, its alpha matte and the video example were provided by the authors. The subjects designed each animation watching its preview, and the final long sequences rendered offline by the authors were shown in the supplemental video. Fig. 12 shows the thumbnails of the animations designed by the users and the time to complete each animation.

During this informal study, we gave explanations and advices, when we found the test users required them. We also asked them to give us feedback in real time during the design process. Design of orientation component seems intuitive even for novices, but as for speed component, the relationship between a specified speed value and its effect in the final animation seems difficult for novices to estimate. The approximated average image and its related parameters also seem not intuitive for them, until they experience several times of trials and errors. However, of such difficulties, the test users could design flow animations in relatively short time as shown in Fig. 12 and the supplemental video. Through this informal study, we have shown that it is possible even for the novices to create flow animations using our system, after a certain amount of training.

Quality Discussion We have shown that our method enables the user to create interesting fluid animations. How-

Scene	Res	OB	SB	CB	w_f	τ	*Sh* (msec)	w_a	HM Grid ($x \times y$)	HM Rep
River	254×176	6	6	0	12	16	0.1	8	4×3	3
Campfire	360×480	6	3	2	40	16	0.4	16	4×4	3
Water Surface	500×208	2	3	0	20	16	0.2	10	8×4	3
Brook	288×384	29	5	3	30	16	0.1	8	5×7	3
Waterfall-I	405×399	21	7	6	30	16	0.1	4	8×8	3
Waterfall-II	617×224	10	5	2	30	16	0.2	8	10×4	3
Smoke-I	283×307	8	2	3	40	16	0.27	14	8×8	3
Smoke-II	236×314	8	2	11	30	16	0.33	14	4×7	3

Table 1: *"Res" shows the resolution of the resulting flow animation. "OB", "SB" and "CB" count the numbers of used Orientation Brushes, Speed Brushes and Clone Brushes. Each parenthetic number shows the number of brushes painted in local mode. w_f is the width of a Gaussian kernel for controlling the sensitivity of the refined flow field to the image gradients. τ is the life time for the video reconstruction. Sh and w_a are the shutter speed and the width of a Gaussian kernel for approximating the average image. "HM Grid" shows the grids dividing the image space for histogram matching. "HM Rep" shows the number of iterations of histogram matching. "River" is shown in Fig. 2-j. "Campfire" and "Water Surface" are shown in Fig. 1.*



(a) 7 min (b) 17 min (c) 11 min

Figure 12: *User experience and editing time.*

ever, there are noticeable artifacts found in some of the resulting animations. We describe the quality problems here.

One of the problems is artifacts like shower door effect, which are significantly seen in the fire animation of our user experience (Fig. 12-a). Since the subject applied only a small amount of blur to the average image, the original fire shapes of the target image are stiffly preserved in the final animation. To prevent this problem, an adequate choice of a set of blur parameters is important.

Another problem is the synthesized animation tends to have viscosity higher than the video example has. Such artifacts are noticeable in "Brook" and the cascade animation of our user experience (Fig. 12-b). There are three reasons. First reason is histogram matching. When the target image has an appearance smoother than the video example, histogram matching reduces high frequency that the video example has. Second reason is a roughly designed flow field. In "Brook" animation, there are small fluid features around the center. Since the flow field is roughly designed using a large w_f parameter, the resulting animation does not capture such small features, resulting in a high-viscosity animation. The other reason is poorly transferred residuals. As we have discussed in Section 5.2, if the residuals are transferred ignoring the relationship between the residuals and the flow field, the resulting animation tends to be random noises, which often look smoother or noisier than the video example.

The white smoke of "Smoke-II" looks little noisy. Strong features appear in areas where more flatness is desired. This implies that the global adjustment of contrast and brightness in Eq. 8 is not enough. We plan to introduce spatially varying adjustment of contrast and brightness, which adaptively changes strengths of fluid features in the image space.

Our method is useful for synthesizing continuous flow animations, but it is difficult to animate a dynamic scene like an ocean surf, whose optical flow is changing drastically. This is beyond our current scope of roughly stationary temporal dynamics. We plan to explore the extensibility of the current technique to such drastically changing fluid motions.

7. Conclusion and Future Work

We have proposed a method to synthesize a continuous flow animation by combining a still fluid image specifying the desired appearance and a video example specifying the desired fluid motion. We abandon physics-based reality but employ the user's sense and imagination based on the observation that a human can easily imagine a rough motion in a fluid picture and identify similarities between a fluid picture and a video example. We developed painting interfaces to convey their ideas to the computer in such an intuitive way as illustrated in this paper. We have demonstrated that our method is useful to quickly and easily design a continuous flow animation of various types of fluid pictures. However, animating a fluid picture is still a challenging problem. We certainly recognize that our solution is still far from the goal, since it requires a certain amount of user interactions and noticeable artifacts are found in a few of our resulting animations. We nevertheless believe that our approach takes a small step toward the goal.

One of our future work is to propose a more justifiable

method, since we have not provided any theoretical underpinnings to justify the quality of the resultant animations. We believe the overall framework of the video analysis/synthesis shown in Fig. 2 holds some technical soundness, but we admit each component in the video synthesis process is still ad hoc. This is because our problem is ill-posed and each component relies on the user interactions. To relax the ill-posedness, we plan to investigate a hybrid approach based on physical model and video example in the future.

Automation of our method is one of the future directions. This includes flow field estimation from a still fluid image and improved automatic transfers. We also plan to build a large fluid video database and provide an automatic search of video examples suitable for the given target image. We also plan to explore the extensibility of the current technique to other types of roughly stationary temporal dynamics like animations of tree swaying.

Acknowledgements

We'd like to thank Karol Myszkowski, Robert Strzodka, Thorsten Grosch, and Miguel Granados for their detailed comments and quick and accurate proofreading. We'd also like to thank Katsumi Takao for providing the campfire painting. We also thank the Eurographics 2009 reviewers for their thoughtful comments. This work is partially supported by JSPS (Japan Society for the Promotion of Science) Research Fellowship and JST (Japan Science and Technology Agency), CREST project.

References

[BC02] BARRETT W. A., CHENEY A. S.: Object-based image editing. In *Proc. of SIGGRAPH 2002* (2002), pp. 777–784.

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

[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.

[Bou99] BOUGUET J.-Y.: Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. *In Intel Research Laboratory, Technical Report* (1999).

[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.

[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. of SIGGRAPH 2005* (2005), pp. 853–860.

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

[FAH91] FREEMAN W. T., ADELSON E. H., HEEGER D. J.: Motion without movement. In *Proc. of SIGGRAPH '91* (1991), pp. 27–30.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. of SIGGRAPH 2001* (2001), pp. 23–30.

[FM96] FOSTER N., METAXAS D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483.

[FP02] FORSYTH D. A., PONCE J.: *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *Proc. of SIGGRAPH '95* (1995), pp. 229–238.

[HE04] HAYS J., ESSA I.: Image and video based painterly animation. In *Proc. of NPAR '04* (2004), pp. 113–120.

[HJN03] HASHIMOTO R., JOHAN H., NISHITA. T.: Creating Various Styles of Animations Using Example-Based Filtering. In *Proc. of Computer Graphics International 2003* (2003), pp. 312–317.

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

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. In *Proc. of SIGGRAPH 2005* (2005), pp. 1134–1141.

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

[LK81] LUCAS B., KANADE T.: An iterative image registration technique with an application to stereo vision. *In Proc. of the International Joint Conference on Artificial Intelligence* (1981), 674–679.

[LW94] LITWINOWICZ P., WILLIAMS L.: Animating images with drawings. In *Proc. of SIGGRAPH '94* (1994), pp. 409–412.

[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.

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *Proc. of SIGGRAPH 2004* (2004), pp. 449–456.

[NKL*07] NARAIN R., KWATRA V., LEE H.-P., KIM T., CARLSON M., LIN M.: Feature-guided dynamic texture synthesis on continuous flows. In *Proc. of Eurographics Symposium on Rendering 2007* (2007).

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. In *Proc. of SIGGRAPH 2000* (2000), pp. 465–470.

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

[SWHS97] SALISBURY M. P., WONG M. T., HUGHES J. F., SALESIN D. H.: Orientable textures for image-based pen-and-ink illustration. In *Proc. of SIGGRAPH '97* (1997), pp. 401–406.

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. *ACM Trans. Graph. 22*, 3 (2003), 716–723.

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

[WZ03] WANG Y., ZHU S.-C.: Modeling textured motion: Particle, wave and sketch. In *Proc. of Int'l Conf. on Computer Vision '03* (2003), p. 213.