Noname manuscript No.
(will be inserted by the editor)

Animating Pictures of Water Scenes using Video Retrieval

Makoto Okabe · Yoshinori Dobashi · Ken Anjyo

Received: date / Accepted: date

Abstract We present a system to quickly and easily create an animation of water scenes in a single image. Our method relies on a database of videos of water scenes and video retrieval technique. Given an input image, alpha masks specifying regions of interest, and sketches specifying flow directions, our system first retrieves appropriate video candidates from the database and create the candidate animations for each region of interest as the composite of the input image and the retrieved videos: this process spends less than one minute by taking advantage of parallel distributed processing. Our system then allows the user to interactively control the speed of the desired animation and select the appropriate animation. After selecting the animation for all the regions, the resulting animation is completed. Finally, the user optionally applies a texture synthesis algorithm to recover the appearance of the input image. We demonstrate that our system allows the user to create a variety of animations of water scenes.

Makoto Okabe
5th Engineering Bldg., room 619, Shizuoka University, 3-5-1 Johoku, Naka-ku, Hamamatsu, Shizuoka, 432-8561, Japan
Tel.: +81-53-478-1214
Fax: +81-53-478-1214
E-mail: m.o@acm.org

Yoshinori Dobashi
Information Media Environment Lab., Graduate School of Information Science and Technology, Hokkaido University, Kita-ku, Kita 14, Nishi 9, Sapporo, 060-0814, Japan
Tel.: +81-11-706-6530
Fax: +81-11-716-8427
E-mail: doba@ime.ist.hokudai.ac.jp

Ken Anjyo
Dens Kono Bldg., room 302, 1-8-8 Wakabayashi, Setagaya, Tokyo, 154-0023, Japan
Tel.: +81-3-3422-3380
Fax: +81-3-3422-3380
E-mail: anjyo@acm.org

Keywords single image · interactive design · video database · video analysis/synthesis · fluid animation · texture analysis/synthesis

Fig. 1 We propose a method for creating the animation from a single image of a water scene based on video retrieval. Given an input image (a), our system retrieves videos from a database (b), each of which appears similar to a part of the input image (c). We finally make a composite of the image and the retrieved videos to create the final animation (d).

1 Introduction

Animating parts of a still image has been and still is an active research area in computer graphics, and creating natural animations of water remains a great challenge. Several attempts to do this have been made but they have not been very successful, since the methods used are limited to periodic motion or are computationally very expensive [3,20,5,21]. Furthermore, the results obtained using these previous methods still look unnatural when compared to real footage. Creating natural, high-resolution animation of water from a single image has not yet been achieved.

We assume that the unnaturalness comes from the fact that these previous methods use only a small amount of the information contained in an actual fluid flow. In Chuang et al.'s method [3], only subtle periodic fluctuations on the water surfaces could be produced since only stochastic information about the fluid motion was used. In Okabe et al.'s method [20] and Gui et al.'s method [5], they failed to produce a wide variety of fluid motions since only a single video example was used. Using local information, such as small patches, is a nice approach [29,21]; however, in this method, information on the global structure of the fluid flow was lacking. In this paper, in order to address the problems encountered in the prior work, we present a new method for water animation in a picture. We collect as many video examples as possible and try to use them as they are. Our method is easy to parallelize and provides the user with quick feedback. This allows us to create various animations from single input pictures.

The basic idea of our method is to use a large video database of real water scenes and replace water regions in the input image with those in the videos. The key to the success of creating natural animation is to use coherent video regions as much as possible. More concretely, for each of the water regions with similar appearance in the input image, our method finds the best matching region from all the video examples: the user is asked to specify a set of coherent regions in the input image with similar appearance and their overall

flow directions; then our method automatically finds the best matching video for each of the user-specified regions by registration. This simple approach produces high quality results as shown in Figure 1 and the supplementary video.

The benefits of our method are as follows. Firstly, we can create high quality and high resolution results that cannot be achieved by any of the previous methods. Secondly, since we use a video database, the user can avoid spending time on manually searching for suitable videos. Thirdly, the method is very simple and is suitable for parallelization. This results in a short turn around time, allowing the user to try different results interactively by selecting candidate animations and specifying the desired speed of the animation.

Each technique involved in our system, e.g., video retrieval and synthesis, is intuitive and has no substantial new technical contribution. However, our system and application are novel: there has never been a system like ours, where the resulting animations are compelling, and a good speedup is obtained over prior work.

Our method of video synthesis performed by a form of two-band blending (Sec. 3.4) is simple but does not work well for very dynamic motions of fire and smoke, which change their overall shapes dynamically over time. Therefore our current scope is limited to water scenes, where inside texture dynamically changes but overall shapes of water are almost static over time. In other words, our simple blending method works surprisingly well for stationary temporal dynamics like water animations. We demonstrate the power of this new approach by showing several examples of water animation in real images as well as paintings.

2 Previous work

Video texture synthesis creates animations of fluids from video examples [29, 1, 27, 4]. These methods, however, do not allow the user to modify the appearance or motion of the synthesized animation. Wang and Zhu [28] analyzed fluid animation, represented it with textons, and synthesized an animation. Bhat et al. [2] proposed a sketching interface that enables users to edit a fluid animation in a video example. 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. [12] developed a method by which animation of a texture flowing over a user-specified motion field can be designed, but where only a stationary motion field was demonstrated without any high-frequency fluid features. Ma et al. [19] extended example-based texture synthesis enabling users to affect the motion field details. *Cliplets* allows users to edit an input video to create a cinemagraph, where still image and video segments are juxtaposed [11]. These methods enable users to modify an existing animation and to synthesize a novel animation, but they do not allow users to specify just a single image for the animation.

Many methods have been proposed for creating animations from a single image [9,10,8,31]. These methods are useful for touring into a picture and creating character animations, but fluid animation is beyond their scope. Chuang et al. [3] proposed a method to synthesize an animation with stochastic motion from a single image, but this technique only supports subtle oscillation of the water surface, such as ripples, and not flowing fluid animation. Lin et al. [16] synthesized animation from multiple, instead of single, high-resolution stills.

Our method is related to image and video database, which are recently used to image and video syntheses for computer graphics applications. Millions of photographs are useful for scene completion [6]. SIFT Flow is used to estimate the motion field in a single image and also to transfer part of a video example onto a single image [17]. A video database of outdoor scenes is useful for scene relighting [14]. A database of time-lapse videos is useful to create an image at a different time of day from an input image [26]. Laffont et al. demonstrated that manually annotated thousands of images from webcams are useful to modify transient attributes in outdoor scenes [13]. However, these methods don't address the problem of animating a picture of fluid using a video database.

Our method is closely related to those synthesized fluid animations that use examples of videos [20,5,21,22]. These methods use a still image for the input as a guide to synthesize the animation. Okabe et al. [20] and Prashnani et al. [22] used a single video for animating a picture of a fluid. In this method, however, the quality of results is highly dependent on the choice of video. Okabe et al. improved this approach by utilizing a video patch database [21]. Although this method has the potential to create high quality animations, its computational cost is prohibitively expensive. The method is also inherently difficult to accelerate by parallel computation. Thus, the method is not easily scalable and it is difficult to create the high resolution animation often required in a production environment. Furthermore, the method often creates incoherent fluid flows, resulting in unnatural animation, since it focuses on the matching of a local flow field within a patch to find the best patch.

3 Our approach

The inputs to our system consist of three parts. Figure 2 shows the input image that the user wants to animate, alpha masks specifying the regions of interest, and sketches specifying the flow directions. Given these inputs, our system retrieves videos of water scenes from a database (Section 3.2) so that the appearance and motion of part of each video matches each user-specified region of interest in the input image. Our system retrieves the 16 best-matched videos for each region of interest (Section 3.3), and shows the user candidate animations of composites of the input image and each retrieved video (Section 3.4). Figure 3 shows four versions of composites for each region of interest: one region is a waterfall and the other region is a river. The user then observes the candidate animations and manually selects the most appropriate

one for each region. Our user interface is inspired by Hays et al.'s method [6]. Finally, the system makes a composite of the input image and the selected videos to create the animation. Furthermore, the user can optionally transfer features from the input image to the animation so that the texture of the resulting animation becomes similar to the input image.

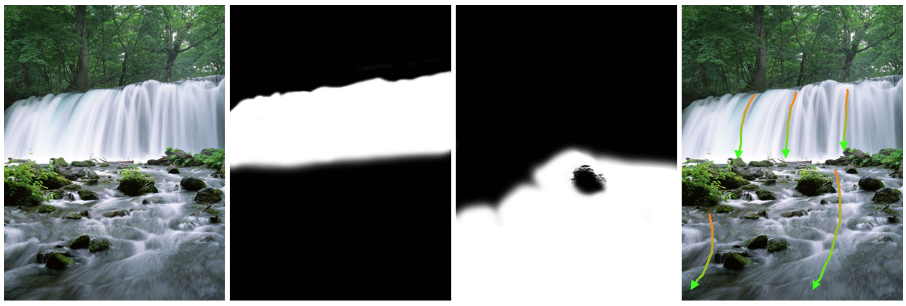

Fig. 2 The input image, the user-specified regions of interest (consisting of the waterfall region and the river region), and the user-specified flow directions.

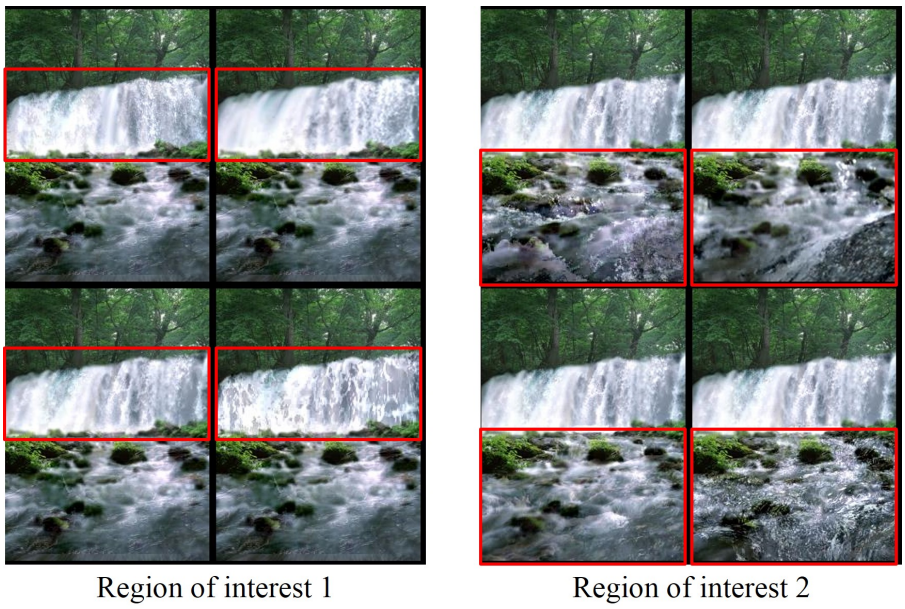

Fig. 3 Four versions of composites made using videos retrieved for the waterfall region (left) and the river region (right).

3.1 Problem Definition

Figure 4 illustrates our method for animating pictures of water scenes. Figure 4-a shows the input image. Figure 4-b shows the part of the waterfall specified by the user as the region of interest. We retrieve the video shown in Figure 4-c from the database and overlap the waterfalls in the input image and the video as shown in Figure 4-d. Since both waterfalls have similar appearance and would have similar motion, we can animate the input image by making the composite of the input image and the video.

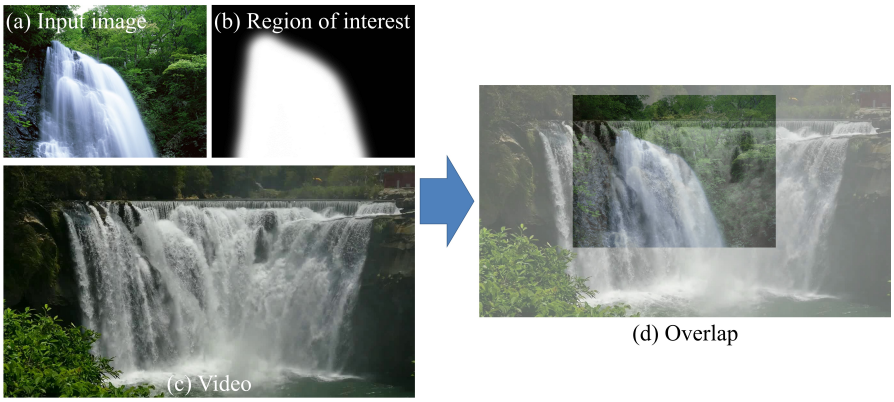

Fig. 4 Illustration of our method for animating pictures of water scenes.

To create a high-quality animation based on this idea, we propose four criteria: 1) the intersection between the region of interest in the input image and the water parts in the video should be as large as possible; 2) the input image and the video should have similar appearance; 3) the input image and the video should have similar flow directions; 4) the motion in the video should match the motion expected from the input image. We define a similarity function to mathematically formulate these criteria:

$$S = S^{overlap}(i, \mathbf{t}) \times \tag{1}$$

$$\sum_{\mathbf{p}} S^{feature}(\mathbf{p}, i, \mathbf{t}) \times S^{flow}(\mathbf{p}, i, \mathbf{t}) \times S^{motion}(\mathbf{p}, i, \mathbf{t}) \times S^{mask}(\mathbf{p}, i, \mathbf{t}), \tag{2}$$

where the four functions $S^{overlap}$, $S^{feature}$, S^{flow}, and S^{motion} correspond to the four criteria, respectively. Let $\mathbf{I}$ denote the input image. $\mathbf{p} = (p_x, p_y)$ represents the pixel position in $\mathbf{I}$. i represents the index of a video in the database. $\mathbf{t}$ represents a two-dimensional (2D) vector $\mathbf{t} = (t_x, t_y)$. The ranges of $\mathbf{t}$ are $-w^{\mathbf{I}} < t_x < w^{\mathbf{I}}$ and $-h^{\mathbf{I}} < t_y < h^{\mathbf{I}}$, where $w^{\mathbf{I}}$ and $h^{\mathbf{I}}$ are the width and height of the input image $\mathbf{I}$.

Our procedure is as follows: first, we maximize the function S for the r-th region of interest with respect to the index of the video i and its translation $\mathbf{t}$,

i.e., $\arg\max_{i,\mathbf{t}} S$. Then, we create an animation of the r-th region of interest by making a composite of the input image and the i-th video translated by $\mathbf{t}$. Note that S is defined not as weighted sum of $S^{overlap}$, $S^{feature}$, S^{flow}, and S^{motion} but as multiplication of them: we must satisfy all the constraints of $S^{overlap}$, $S^{feature}$, S^{flow}, and S^{motion}; in other words, if any one of them are lacking, it will not produce any satisfactory result. However, since the direct maximization of S is difficult, we create candidate animations in terms of S (as shown in Fig. 3) and ask the user to select one of them.

We define the first criterion about intersection $S^{overlap}$ as follows:

$$S^{overlap}(i, \mathbf{t}) = A(\mathbf{M}_r \cap T(\mathbf{M}^{\mathbf{V}_i}, \mathbf{t})). \tag{3}$$

Let $\mathbf{A}_r$ denote the user-specified alpha mask representing the r-th region of interest. We define a binary mask $\mathbf{M}_r$ as follows:

$$\mathbf{M}_r(\mathbf{p}) = \begin{cases} 1 & \|\mathbf{A}_r(\mathbf{p})\| \geq t^{mask}, \\ 0 & otherwise. \end{cases} \tag{4}$$

$A(\mathbf{M})$ is a function that computes the area of a binary mask $\mathbf{M}$. $T(\mathbf{M}, \mathbf{t})$ is a function that translates a binary mask $\mathbf{M}$ using $\mathbf{t}$. Let $\mathbf{V}_i$ and $\mathbf{M}^{\mathbf{V}_i}$ denote the first frame of the i-th video in the database and the binary mask representing the water parts in the frame, respectively.

We define the second criterion $S^{feature}$ that the appearance of the input image and the video should be similar as follows:

$$S^{feature}(\mathbf{p}, i, \mathbf{t}) = s^{feature}(f(\mathbf{p}, \mathbf{I}), f(\mathbf{p} - \mathbf{t}, \mathbf{V}_i)). \tag{5}$$

The function f is a SIFT feature descriptor [18]: $f(\mathbf{p}, \mathbf{I})$ computes the feature vector of $\mathbf{I}$ at the pixel position $\mathbf{p}$. Given a pair of feature vectors, $s^{feature}$ computes the similarity between the appearances.

We define the third criterion of similarity of the flow directions S^{flow} as follows:

$$S^{flow}(\mathbf{p}, i, \mathbf{t}) = s^{flow}(\mathbf{F}(\mathbf{p}), \mathbf{F}^{\mathbf{V}_i}(\mathbf{p} - \mathbf{t})). \tag{6}$$

Let $\mathbf{F}$ denote the flow field made from the user-specfied flow directions, and let $\mathbf{F}^{\mathbf{V}_i}$ denote the average flow field through video frames. Given a pair of 2D vectors for the flow directions, s^{flow} computes the similarity between them.

We define the fourth criterion about motion of water S^{motion} as follows:

$$S^{motion}(\mathbf{p}, i, \mathbf{t}) = s^{motion}(m_{\mathbf{p}}(\mathbf{I}), m_{\mathbf{p}-\mathbf{t}}(\mathbf{V}_i)). \tag{7}$$

$m_{\mathbf{p}}(\mathbf{I})$ represents a function that estimates the motion of $\mathbf{I}$ at the pixel position $\mathbf{p}$. Note that $m_{\mathbf{p}}(\mathbf{I})$ represents fluid motions that cannot be described by $\mathbf{F}(\mathbf{p})$. $\mathbf{F}(\mathbf{p})$ represents only flow directions, but fluid motions involve the other elements like flow speeds, variances, and frequencies of appearing and disappearing water drops, etc. Here, we let $m_{\mathbf{p}}(\mathbf{I})$ denote all these elements.

We define S^{mask} as follows:

$$S^{mask}(\mathbf{p}, i, \mathbf{t}) = \mathbf{M}_r(\mathbf{p})\mathbf{M}^{\mathbf{V}_i}(\mathbf{p} - \mathbf{t}), \tag{8}$$

which guarantees that similarity measurements are performed within the masks.

The direct maximization of S is difficult: it is possible to compute $S^{overlap}$, $S^{feature}$, S^{flow}, and S^{mask}, because all the functions in them are well-defined and all the required information is provided as the user's inputs; however, the function m in S^{motion}, which estimates the fluid motion only from a single image, is unknown and difficult to implement. Therefore, our strategy for maximizing S is to compute S', which consists of $S^{overlap}$, $S^{feature}$, S^{flow}, and S^{mask}, for all the possibilities of i and $\mathbf{t}$. We then show the user the candidate animations, i.e., a set of animations made using the i-th video translated by $\mathbf{t}$ that gives high similarity values of S'. We complete the maximization of S by asking the user to select the animation, where the user feels that S^{motion} is maximized, i.e., the fluid motion of the animation is best matched to the fluid motion expected from the input image.

Note that our purpose is to create animations such as the waterfall in Figure 4, which have time-varying appearance but roughly stationary temporal dynamics [4]. Since the overall appearance of such a scene does not change dramatically with time, we can use a single representative image to describe the appearance of a video.

Both $S^{overlap}$ and S^{mask} are required in Eqs. 1– 2. These two terms may look redundant: these are similar in the sense that both compute the overlap between the input image and a video. However, these are different: $S^{overlap}$ computes the overlapping area, and Eq. 2 computes the number of pixels satisfying the criteria. In other words, when multiple pairs of i and $\mathbf{t}$ give the same value of Eq. 2, we want to create animations using pairs that give a larger overlap of $S^{overlap}$.

3.2 Video database

We gathered 202 videos of water scenes filmed at various locations in the real world, e.g., from Niagara Falls to unknown small cascades. The resolution of each video in the database is 1280×720 pixels. Thumbnails of all our videos are included in the supplementary material. In each video, the camera is fixed and focused on the water itself, and no other significant moving objects exists other than the water itself. We prepared the horizontally flipped version of each video to double the number of videos without any loss of the quality of the database. Given the database of videos, we compute the flow field $\mathbf{F}^{\mathbf{V}_i}$, the mask $\mathbf{M}^{\mathbf{V}_i}$, and the alpha matte $\mathbf{A}_{i,t}$ for the i-th video in the database.

We apply the optical flow method [30] to each pair of neighboring video frames to compute the 2D velocity vector at each pixel. Since the optical flow method often gives noisy or erroneous velocities, especially for fast running water, we take the average of velocities at each pixel through the duration of 30 video frames, to obtain a smooth flow field $\mathbf{F}^{\mathbf{V}_i}$. Since the videos in the database and also our target animation are roughly stationary temporal dynamics [4] whose overall appearance does not change dramatically with

time, we assume that such a short video clip as 30 video frames are enough to estimate good average velocities in a video.

Since we want to create an animation using only water but not other objects, e.g., rocks and trees around the water, we compute the mask $\mathbf{M}^{\mathbf{V}_i}$ to extract the water parts. Since no significantly moving objects exists other than water in each video, we can make the mask based on the magnitude of the flow field:

$$\mathbf{M}^{\mathbf{V}_i}(\mathbf{p}) = \begin{cases} 1 & \|\mathbf{F}^{\mathbf{V}_i}(\mathbf{p})\| \geq t^{mask}, \\ 0 & otherwise, \end{cases} \tag{9}$$

where we use 2.0 for t^{mask} in all our experiments. Figure 5 shows a video and its mask.

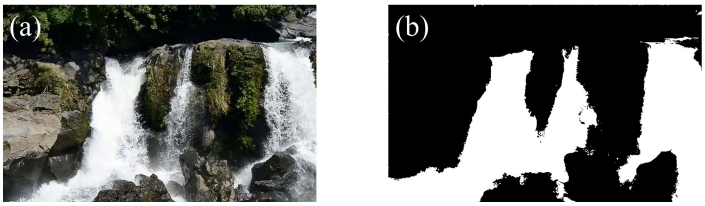

Fig. 5 (a) A video frame. (b) The mask $\mathbf{M}^{\mathbf{V}_i}$.

We compute the alpha matte $\mathbf{A}_{i,t}$ for the t-th video frame, which is used during the animation synthesis (Section 3.4). We compute the alpha matte based on a color histogram model [24]. Let P denote the set of pixels positions $\mathbf{p}$ where $\mathbf{M}^{\mathbf{V}_i}(\mathbf{p})$ is one. Let $\overline{P}$ denote the set of the other pixel positions. We assume that P represents the water parts and $\overline{P}$ represents the other objects around the water. We compute a color histogram for P and $\overline{P}$. Each pixel in a video frame has a color, i.e., a three-dimensional (3D) red-green-blue (RGB) vector $\mathbf{c} = (c_r, c_g, c_b)$, where the range of each element is $0 \leq c < 256$. We map $\mathbf{c}$ to the bin of a color histogram of B^3 dimensions using the following function: $b(\mathbf{c}) = (\lfloor \frac{c_r}{256/B} \rfloor, \lfloor \frac{c_g}{256/B} \rfloor, \lfloor \frac{c_b}{256/B} \rfloor)$. We compute the color histogram through all the video frames: let H_P and $H_{\overline{P}}$ denote the color histograms computed using the set of pixel positions P and $\overline{P}$ respectively. Finally, we compute the alpha matte $\mathbf{A}_{i,t}$ as the probability: $\mathbf{A}_{i,t}(\mathbf{x}) = \frac{H_P(b(\mathbf{c}))}{H_P(b(\mathbf{c}))+H_{\overline{P}}(b(\mathbf{c}))}$, where $\mathbf{x}$ represents the pixel position with color $\mathbf{c}$. Figure 6-b shows the resulting alpha matte, where the details around the boundaries are successfully captured. We use 4 for B in all our experiments.

3.3 Video retrieval

We find the set of i and $\mathbf{t}$ that gives a high similarity value of S'. Our strategy is an exhaustive search. We evaluate S' for all the possibilities of i and $\mathbf{t}$.

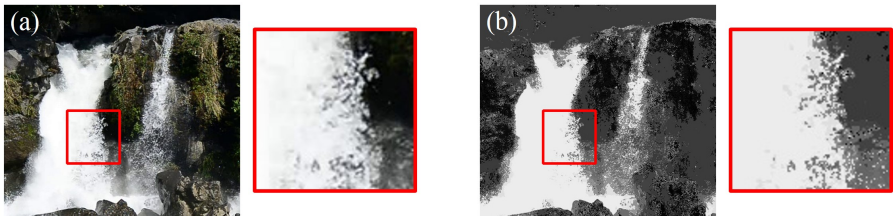

Fig. 6 (a) A video frame. (b) Our alpha matte.

We first consider feature points in regular grids of the input image and also the video (Figure 7). The intervals between adjacent feature points are 16 pixels in all our experiments. We introduce these sparse regular grids, because we want to retrieve videos not by exact but by rough feature matching: please note that our purpose is not to reproduce the ground truth motion from a single image but to create an animation using the input image as a reference. As shown in Figure 7, we consider feature points only at $\mathbf{x}$, where $\mathbf{M}_r(\mathbf{x})$ or $\mathbf{M}^{\mathbf{V}_i}(\mathbf{x})$ is one. Therefore, since we consider these feature points and also, from now on, the translation $\mathbf{t}$ with a 16 pixel interval, we assume that the term of S^{mask} has already been maximized. The evaluation of $S^{overlap}$ is the same as counting the number of feature points $\mathbf{x}$ that satisfy both $\mathbf{M}_r(\mathbf{x})$ and $T(\mathbf{M}^{\mathbf{V}_i}, \mathbf{t})(\mathbf{x})$ are one.

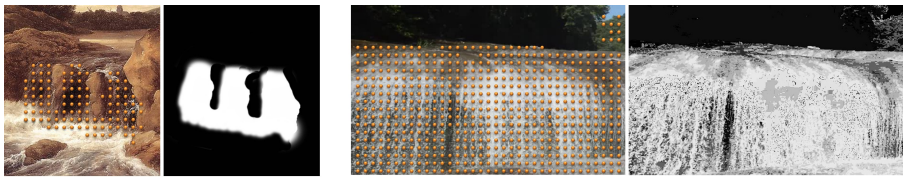

Fig. 7 We consider feature points in sparse regular grids.

As for the second criterion $S^{feature}$, we consider an image patch of 64×64 pixels around each feature point. Our image feature is based on a histogram of the image gradients, which is similar to the feature description by a scale-invariant feature transform (SIFT) without the scale and rotation invariance [18]: we divide a image patch into 4×4 blocks and compute the histogram of the image gradients with 8 bins corresponding to 8 directions. The resulting 128 dimensional feature vector is normalized so that the summation of all bins is equal to 1.0. Given a pair of feature vectors, $\mathbf{f}^{\mathbf{I}}$ and $\mathbf{f}^{\mathbf{V}}$, our similarity function in $s^{feature}$ is as follows:

$$s^{feature}(\mathbf{f}^{\mathbf{I}}, \mathbf{f}^{\mathbf{V}}) = \begin{cases} 1 & \|\mathbf{f}^{\mathbf{I}} - \mathbf{f}^{\mathbf{V}}\| < t^{feature}, \\ 0 & otherwise, \end{cases} \tag{10}$$

where we use 0.08 for $t^{feature}$ in all our experiments.

As for the third criterion S^{flow}, we compare the flow directions at corresponding pixels in the input image and the video. As for the input image, the user sketches the desired flow directions sparsely (Figure 2-rightmost). Therefore, we compute a dense flow field by interpolating the sparse information. The flow directions are specified along the user-drawn arrows, we interpolate them to obtain $\mathbf{F}$ using the radial basis function $\phi(\mathbf{x}) = |\mathbf{x}|$ [23]: we interpolate the x- and y-components of the flow directions separately. As for the video, we use the flow field $\mathbf{F}^{\mathbf{V}_i}$. Given a pair of 2D flow direction vectors, $\mathbf{d}^{\mathbf{I}}$ and $\mathbf{d}^{\mathbf{V}}$, our similarity function in s^{flow} is as follows:

$$s^{flow}(\mathbf{d}^{\mathbf{I}}, \mathbf{d}^{\mathbf{V}}) = \begin{cases} 1 & \frac{|\mathbf{d}^{\mathbf{I}} \cdot \mathbf{d}^{\mathbf{V}}|}{|\mathbf{d}^{\mathbf{I}}||\mathbf{d}^{\mathbf{V}}|} > t^{flow}, \\ 0 & otherwise, \end{cases} \tag{11}$$

where we use $\cos(\frac{\pi}{6})$ for t^{flow} in all our experiments.

We rely on the user's sketch to obtain the water flow directions of an input image, because automatic estimation of them is difficult. For example, analysis of image structure tensor allows us to obtain not flow directions but only directions of image edges: even if it is possible to automatically estimate that there is a horizontally flowing river in an image, but it is difficult to know that the river flows toward the left or the right by such a simple image analysis. Since automatic estimation of flow directions requires high-level semantic image analysis, we choose a simple approach of user's sketch.

3.4 Video synthesis

We create an animation of composite of the input image and the retrieved videos. The problem we have to solve is how to combine all the appearances of the mage and the videos. Let $\mathbf{R}_{i,t}$ denote the frame of the final animation, which is initialized by the input image, i.e., $\mathbf{R}_{i,t} = \mathbf{I}$ for all the t-th video frame.

Motion in a typical water scene, e.g., waterfalls and rivers, has roughly stationary temporal dynamics: the overall appearance and overall shape of water do not change dramatically but small details such as water splashes are moving. Our solution is to create the overall appearance and overall shape of the animation using the low frequency component of the input image and move small details using the high frequency component of the videos. Figure 8 shows the process. We apply Gaussian blur to the input image $\mathbf{I}$ (Figure 8-a) and obtain a blurry image $\mathbf{I}^B$ (Figure 8-b) that represents the low frequency component: let s^I denote the size of the Gaussian kernel. We also apply Gaussian blur to the t-th video frame $\mathbf{V}_{i,t}$ (Figure 8-d) and obtain a blurry version $\mathbf{V}^B_{i,t}$ (Figure 8-e): let s^V denote the size of the Gaussian kernel. We obtain the high frequency residuals $\mathbf{V}^R_{i,t}$ (Figure 8-f) by the following subtraction: $\mathbf{V}^R_{i,t} = \mathbf{V}_{i,t} - \mathbf{V}^B_{i,t}$. We make the composite $\mathbf{V}^C_{i,t}$ by the following addition: $\mathbf{V}^C_{i,t}(\mathbf{p}) = \mathbf{I}^B(\mathbf{p}) + \mathbf{V}^R_{i,t}(\mathbf{p} - \mathbf{t})$. This method is simple but powerful to success-

fully make the realistic composite of the input image and the video. We use 15 and 63 for s^I and s^V respectively in all our experiments.

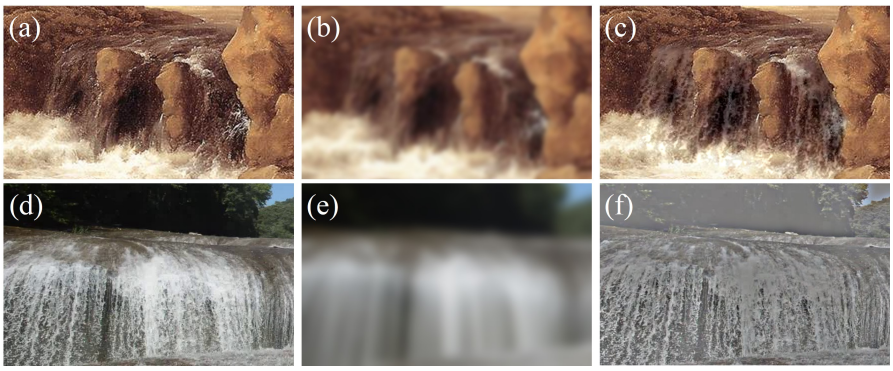

Fig. 8 The process of making the composite of the input image and the video.

Finally, we extract the region of interest from $\mathbf{V}^C_{i,t}$ and make the composite with $\mathbf{R}_{i,t}$. We make the alpha matte $\mathbf{A'}_{i,t}$ as the following multiplication: $\mathbf{A'}_{i,t}(\mathbf{p}) = \mathbf{A}_r(\mathbf{p})\mathbf{A}_{i,t}(\mathbf{p} - \mathbf{t})$. We then update the animation by applying alpha blending: $\mathbf{R}_{i,t}(\mathbf{p}) = \mathbf{V}^C_{i,t}(\mathbf{p})\mathbf{A'}_{i,t}(\mathbf{p}) + \mathbf{R}_{i,t}(\mathbf{p})(1 - \mathbf{A'}_{i,t}(\mathbf{p}))$. Figure 8-c shows the resulting composite, where the alpha matte is also applied to affect only the waterfall region. We repeat the process for all the regions of interest in the input image.

Smooth transition between neighboring regions is dependent on the three factors: 1) smooth boundaries of the user-specified alpha mask A_r, 2) the success of the video retrieval, and 3) manual selection of videos by the user. We expect that the user paints the alpha mask using brush tools in a standard image editing software, i.e., the region of interest A_r has smooth, blurry boundaries as shown in Fig. 2 or Fig. 4. We can expect that the video retrieval described in Sec. 3.3 successfully retrieves videos whose appearance and motion are consistent with the input image and user's sketch. We also expect that the user has chosen an appropriate set of videos for all the regions during the manual selection described in Sec. 3 (Fig. 3). Given appropriate alpha masks and videos, the simple equations described above successfully synthesize natural animations with smooth transitions between neighboring regions.

The animation made by the method described above usually looks reasonable but sometimes looks different from the input image. In such a case, we allow the user to apply appearance transfer from the input image to each video frame. Our appearance transfer is the same as the previous method by Okabe el al. [20]. We divide the image space into rectangular patches, where the half part of each patch overlaps its neighboring patch. We apply the texture synthesis method [7] between corresponding patches in the input image and the video frame. We combine all the patches taking weighted average with the

Fig. 9 Effects of appearance transfer.

weight defined by a Gaussian kernel. Figure 9 shows the effects of appearance transfer: the appearances of color and textures become more similar to the input image.

4 Results and discussion

Figure 15 shows video frames of eleven animation produced by using our method. All the results made using a single photograph (Figure 15-a to 15-i) are 640×480 pixel resolution. Figure 15-j and 15-k, which are oil paintings of Jacob van Ruisdael in the 17th century, are 680×878 pixel resolution and 990×704 pixel resolution, respectively.

Given an input image, we start to create the animation by specifying rough alpha masks of regions of interest using Adobe Photoshop CC 2014, where we make a layer for each region of interest and paint it using a standard brush. We then specify flow directions using our interactive tool, where the user sketches strokes over the input image. We usually spend only several minutes to specify rough alpha masks and flow directions.

Our system then retrieves videos for each region of interest: for each i-th video in the database, our system finds the translation $\mathbf{t}$ that gives the highest S', which will be used to create the candidate animation. We use 15 computers, which consist of 100 cores in total, and it takes less than one minute to finish the video retrieval in all our experiments. We simply parallelize the computation of S' for each video in the database.

Our system then renders 16 candidate animations for currently selected region of interest and shows them in 4×4 grids: the animations are placed from top-left to bottom-right in the order of the similarity value S'. The duration of each candidate animation is 24 frames. The user observes each of them and select what the user feels the best animation. When the user feels that speeds of water flow in candidate animations are too fast or too slow, the user can specify the upper or lower limit of the speed. The system then updates candidate animations using only the retrieved videos whose average speed in the region is within the limits. When the user finishes the selection for a region and goes to the next region, the system updates candidate animations. These updates of candidate animations spend just several seconds, because they do not affect the computation of S'. On the other hand, when the user modifies

alpha masks or flow directions, the system requires recalculation of S' and the user has to wait for less than one minute again.

Figure 10 shows a frame of the animation made from the oil painting (Fig. 15-k) with the retrieved video for each region. Although the lighting conditions and colors are different between the oil painting and each video, our simple two-band blending (Sec. 3.4) successfully synthesizes the final animation. Watch the actual animation in the supplementary video.

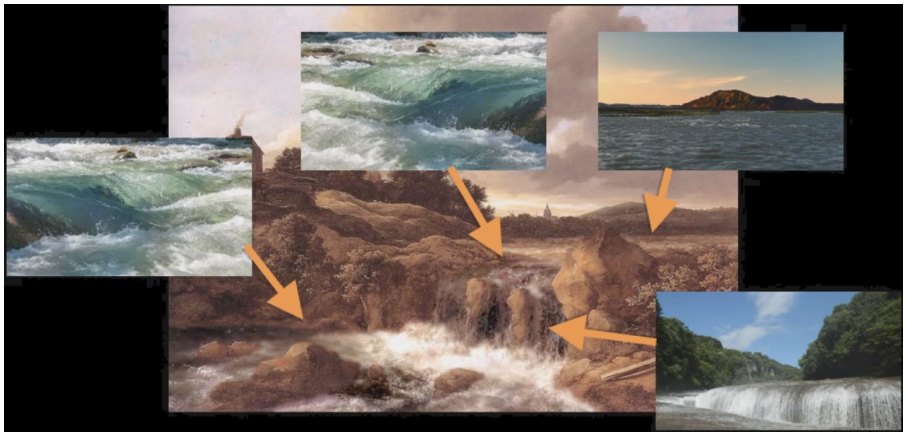

Fig. 10 A frame of the resulting animation with the retrieved videos.

After selecting candidate animations for all regions, the system renders the final animation having a larger number of frames. At this stage, we sometimes need to refine the alpha masks. Especially, we modified alpha masks around rocks and trees to keep the depth-order relations in the animation.

We applied the appearance transfer described in Sec. 3.4 to the results of oil paintings. We did not apply it to the results of photographs: as shown in Fig. 15, photographs of a water scene are usually taken with a slow shutter speed, which results in a beautiful but blurry look. We did not apply the appearance transfer not to transfer the blurry appearance to the resulting animation.

4.1 Subjective evaluation

We performed a subjective evaluation of the visual quality of the resulting animations. The subjects are thirteen students who major in computer science and do not know our research project. We presented the eleven animations (Figure 15). We asked each subject to "rank the visual quality of each animation on a scale ranging from 1 (poorest) to 5 (best)". We also asked to "write a short comment for each animation about visible artifacts, when they were found".

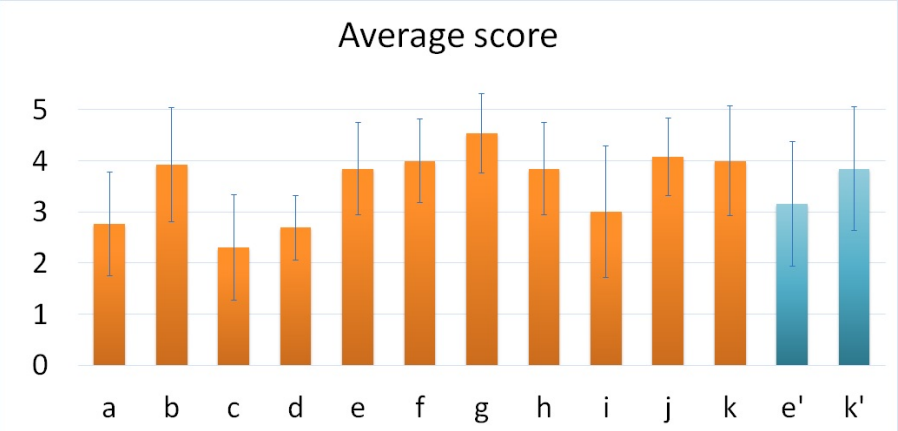

Fig. 11 The result of the subjective evaluation. Each bar represents the average score of visual quality. Each error bar represents the standard deviation. The previous method [21] presents the animations of Figure 15-e and 15-k, which we call *e'* and *k' here.*

The results are shown in Figure 11. The highest average score is assigned to Figure 15-g: multiple subjects wrote comments saying "natural" or "beautiful". The lowest average score is assigned to Figure 15-c: according to the comments, one major reason of low scores is visible block noises, which we can also find in the original retrieved video. All the results of Figure 15, which are used for the subjective evaluation, and the retrieved videos are shown in the supplementary video.

The observers often pointed out the blurriness in the results, e.g., the waterfall in Fig. 15-a. One reason of this blurriness is our two-scale decomposition: we extracted high frequency residuals from videos using the same Gaussian kernel whose size is fixed to s^V in all our experiments (Sec. 3.4). However, ideally, this kernel size should be adaptively changed according to the dominant spatial frequency at each pixel position of a video. We want to investigate this adaptive Gaussian kernel it in the future, where multi-scale analysis/synthesis may give a good solution.

4.2 Comparison

We compare the results of our method with the results of the previous method [21]. The previous method presents the animations of Figure 15-e and 15-k, which we call *e'* and *k'*. We also asked each subject to evaluate them. We obtained the original videos of *e'* and *k'* from the authors of the previous method [21]. Since the resolution of each result of the previous method was low, we scale it up so that the size becomes the same as our result, and we then show it to subjects. The results are shown as the blue bars in Figure 11. Compared with *e* and *e'*, the score of our method is higher, which shows that our method enables to create the animation from the same image with higher visual qual-

ity. Compared with k and k', the score of our method is slightly higher but almost comparable: since multiple subjects reported in their comments that the resolution of k' is low or k' looks blurry compared with our result k, we believe that our result k has better visual quality than k'; however, since subjects seemed not to feel *unconfortable* about the blurriness, the average score of k' did not become low.

We describe the similar and different components of our method compared with the previous method [21]. We construct the database of fluid videos, but the previous method constructs the database of video patches. We retrieve a video for each region of interest, but the previous method retrieves a video patch at each grid point in the input image. The methods of video synthesis are relatively similar: both methods extract high frequency components from each video frame, which are added to the blurred version of the input image. However, while our method applies a large Gaussian kernel to obtain low frequency components (Sec. 3.4), the previous method averages video frames to obtain it. Since such an average image often has a significant amount of high frequency components, the extracted high frequency components look blurry compared with the ones extracted by our method. The appearance transfer described in the last paragraph in Sec. 3.4 is almost the same as the one used in the previous method.

Given an input image, masks, and flow directions, the previous method spent about one hour to create the first animation. On the other hand, our method spends less than one minute. Also, our method enables the user to interactively design the animation by controlling the speed and selecting the best animation from a variety of candidate animations.

It is not easy to create the higher resolution video using the previous method. Figure 12-left shows a frame of k', which is 389×165, i.e., about half resolution of our result. Figure 12-right shows the corresponding retrieved video patches assigned over the image space. We see various video patches from different videos: the waterfall part consists of different waterfall videos; the river part also consists of various river videos and there is no consistency of water flow between neighboring video patches. When we apply the previous method to create a higher resolution video, these inconsistencies show up as significant visible artifacts. Also, a pair of neighboring video patches are overlapped with each other by their half area: they are merged by a smooth blending method during the video synthesis, which decreases the original sharp appearance of each video patch and causes the blurriness. On the other hand, our method produces consistent water flows over each region and enables the creation of higher resolution videos. Figure 13 shows a frame from the animation of our method (left), and assignment of retrieved videos (right).

4.3 Failure cases

Figure 14 shows the input images of typical failure cases. Figure 14-a has the large white part where water bubbles seem to appear. This part is expected

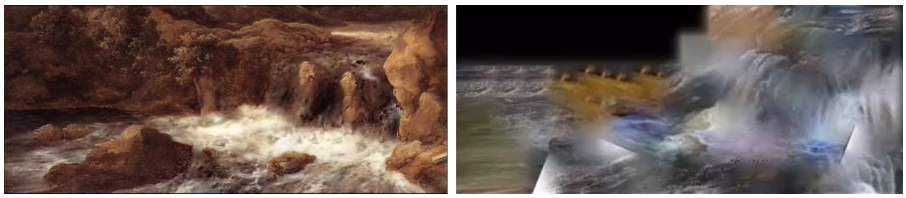

Fig. 12 A frame from the animation of the previous method [21] (left), and its patch-based video retrieval and assignment (right).

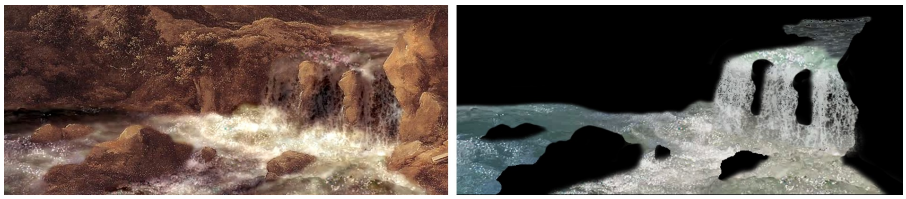

Fig. 13 A frame from the animation of our method (left), and its video retrieval and assignment (right). The retrieved videos are shown in Fig. 10.

to move dynamically. However, they look static under the flowing water in the result. Since the retrieved video does not have such bubbles, our method could not produce dynamic motion in the part. Figure 14-b has small cascades behind the large waterfalls. The waterfalls move but the cascades look static, because the retrieved video has waterfalls without cascades behind.

Fig. 14 Typical failure cases.

5 Limitations and future work

Various types of fluids We have presented animations of only water scenes, but we want to extend our method for animations of fire and smoke in the future.

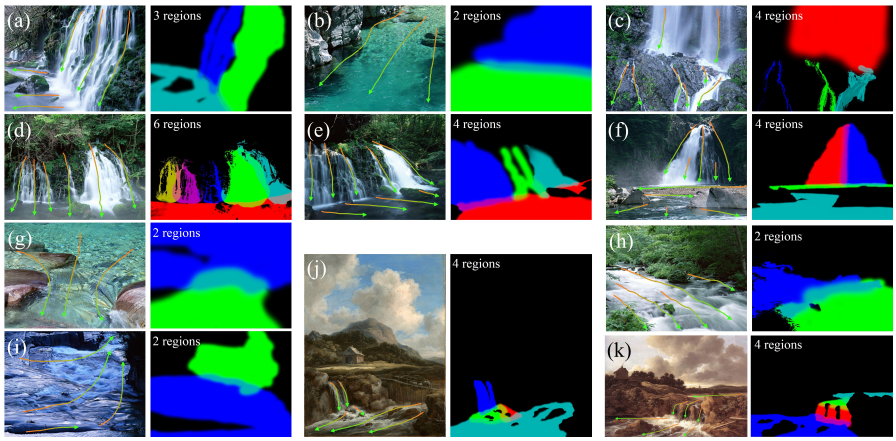

Fig. 15 Each pair of images shows the input image (left), alpha masks specifying regions of interest (right), and user-specified flow directions (arrows on the input image). Actual alpha masks are grayscale images, but we visualize all of them in a single image using different colors. We also show the number of user-specified regions of interest on the image of colored alpha masks.

We currently synthesize an animation as the combination of low frequency component of the input image and high frequency component of the video. However, we cannot use it for dynamic fire and smoke scenes, because low frequency component should also move dynamically in them: for example, the boundaries of fire and smoke would change their shapes with time. We want to investigate the algorithm for animation synthesis that can treat such dynamic motions.

More intelligent system We currently compute S' for all the possibilities of the index of the video and translations. We can extend our method to take more variations into account, e.g., not only translations but also rotations and scales. We may also take not only spatial but also temporal variations by applying time warping, when the retrieved videos are too fast or too slow. We may also extend our method so that it takes variations of alpha masks into account, where the user specifies only a single alpha mask and the system automatically proposes reasonable divisions of it. We may also cope with techniques of semantic scene understanding, which automatically finds fluid parts and their properties in a single image. We investigate such possibilities and techniques to further reduce the user's burden.

Transformations of videos During the video retrieval described in Sec. 3.3, we don't take into account of any transformation of videos in the database, e.g., rotation, scaling, etc. We have not observed any artifacts caused by this so far, but such transformations may improve the efficiency of the video retrieval and

the quality of final animations in exchange for the increase of computational cost. We want to investigate transformations of videos in our future work.

Infinite loop videos It would make sense to incorporate one of the previous methods [25, 2, 15] to create an infinite loop video of a water animation, which is also our future work.

Acknowledgements We would like to thank the anonymous reviewers for their insightful and constructive comments. Many thanks also go to Ayumi Kimura for discussions and encouragements. This work was supported by JSPS KAKENHI Grant Numbers JP15H05924 and JP25730071. This work was supported by Japan Science and Technology Agency, CREST. This work was partially supported by the Joint Research Program (Short-term Collaborative Research) of the Institute of Mathematics for Industry, Kyushu University. Yoshinori Dobashi was partially supported by UEI Research.

References

1. 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), 120–135 (2001). DOI http://dx.doi.org/10.1109/2945.928165
2. Bhat, K.S., Seitz, S.M., Hodgins, J.K., Khosla, P.K.: Flow-based video synthesis and editing. ACM Trans. Graph. **23**(3), 360–363 (2004). DOI http://doi.acm.org/10.1145/1015706.1015729
3. 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, pp. 853–860 (2005)
4. Doretto, G., Chiuso, A., Wu, Y.N., Soatto, S.: Dynamic textures. Int. J. Comput. Vision **51**(2), 91–109 (2003)
5. Gui, Y., Ma, L., Yin, C., Z, C.: Preserving global features of fluid animation from a single image using video examples. Journal of Zhejian University - Science C **13**(7), 510–519 (2012)
6. Hays, J., Efros, A.A.: Scene completion using millions of photographs. ACM Trans. Graph. **26**(3) (2007)
7. Heeger, D.J., Bergen, J.R.: Pyramid-based texture analysis/synthesis. In: Proc. SIGGRAPH '95, pp. 229–238 (1995)
8. Hornung, A., Dekkers, E., Kobbelt, L.: Character animation from 2d pictures and 3d motion data. ACM Trans. Graph. **26**(1), 1 (2007)
9. 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, pp. 225–232 (1997)
10. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM Trans. Graph. **24**(3), 1134–1141 (2005)
11. Joshi, N., Mehta, S., Drucker, S., Stollnitz, E., Hoppe, H., Uyttendaele, M., Cohen, M.: Cliplets: Juxtaposing still and dynamic imagery. In: Proc. of UIST '12, pp. 251–260 (2012)
12. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. In: Proc. SIGGRAPH 2005, pp. 795–802 (2005)
13. Laffont, P.Y., Ren, Z., Tao, X., Qian, C., Hays, J.: Transient attributes for high-level understanding and editing of outdoor scenes. ACM Trans. Graph. **33**(4), 149:1–149:11 (2014)
14. 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), 131:1–131:10 (2009)
15. Liao, Z., Joshi, N., Hoppe, H.: Automated video looping with progressive dynamism. ACM Trans. Graph. **32**(4), 77:1–77:10 (2013)

16. Lin, Z., Wang, L., Wang, Y., Kang, S.B., Fang, T.: High resolution animated scenes from stills. IEEE Trans. Vis. Comput. Graph. **13**(3), 562–568 (2007)
17. Liu, C., Yuen, J., Torralba, A., Sivic, J., Freeman, W.T.: Sift flow: Dense correspondence across different scenes. In: Proc. of ECCV '08, pp. 28–42 (2008)
18. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**, 91–110 (2004)
19. Ma, C., Wei, L.Y., Guo, B., Zhou, K.: Motion field texture synthesis. ACM Trans. Graph. **28**(5), 1–8 (2009)
20. Okabe, M., Anjyo, K., Igarashi, T., Seidel, H.P.: Animating pictures of fluid using video examples. Computer Graphics Forum (Proc. EUROGRAPHICS) **28**(2), 677–686 (2009)
21. Okabe, M., Anjyo, K., Onai, R.: Creating fluid animation from a single image using video database. Computer Graphics Forum (Proc. Pacific Graphics 2011) **30**(7), 1973–1982 (2011)
22. Prashnani, E., Noorkami, M., Vaquero, D., Sen, P.: A phase-based approach for animating images using video examples. Computer Graphics Forum (2016)
23. Praun, E., Finkelstein, A., Hoppe, H.: Lapped textures. In: Proc. SIGGRAPH 2000, pp. 465–470 (2000)
24. Ramanan, D.: Learning to parse images of articulated bodies. In: B. Schölkopf, J. Platt, T. Hoffman (eds.) Advances in Neural Information Processing Systems 19, pp. 1129–1136. MIT Press (2007). URL http://papers.nips.cc/paper/2976-learning-to-parse-images-of-articulated-bodies.pdf
25. Schödl, A., Szeliski, R., Salesin, D.H., Essa, I.: Video textures. In: Proc. SIGGRAPH 2000, pp. 489–498 (2000)
26. Shih, Y., Paris, S., Durand, F., Freeman, W.T.: Data-driven hallucination of different times of day from a single outdoor photo. ACM Trans. Graph. **32**(6), 200:1–200:11 (2013)
27. Sun, M., Su, H., Savarese, S., Fei-Fei, L.: A multi-view probabilistic model for 3d object classes. pp. 1247 –1254 (2009)
28. Wang, Y., Zhu, S.C.: Modeling textured motion: Particle, wave and sketch. In: ICCV2003, pp. 213–220 (2003)
29. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proc. SIGGRAPH 2000, pp. 479–488 (2000)
30. Werlberger, M., Pock, T., Bischof, H.: Motion estimation with non-local total variation regularization. In: Proc. of CVPR 2010, pp. 2464–2471 (2010)
31. Xu, X., Wan, L., Liu, X., Wong, T.T., Wang, L., Leung, C.S.: Animating animal motion from still. ACM Trans. Graph. **27**(5), 117:1–117:8 (2008)