DOI: 10.1111/cgf.12940

COMPUTER GRAPHICS *forum*
Volume 36 (2017), number 6 pp. 303–311

A Phase-Based Approach for Animating Images Using Video Examples

Ekta Prashnani[1], Maneli Noorkami[2], Daniel Vaquero[2] and Pradeep Sen[1]

[1]University of California, Santa Barbara, CA, USA
{ekta, psen}@ece.ucsb.edu
[2]Nokia Technologies, Sunnyvale, CA, USA
{maneli.noorkami, daniel.vaquero}@nokia.com

Abstract

We present a novel approach for animating static images that contain objects that move in a subtle, stochastic fashion (e.g. rippling water, swaying trees, or flickering candles). To do this, our algorithm leverages example videos of similar objects, supplied by the user. Unlike previous approaches which estimate motion fields in the example video to transfer motion into the image, a process which is brittle and produces artefacts, we propose an Eulerian phase-based approach which uses the phase information from the sample video to animate the static image. As is well known, phase variations in a signal relate naturally to the displacement of the signal via the Fourier Shift Theorem. To enable local and spatially varying motion analysis, we analyse phase changes in a complex steerable pyramid of the example video. These phase changes are then transferred to the corresponding spatial sub-bands of the input image to animate it. We demonstrate that this simple, phase-based approach for transferring small motion is more effective at animating still images than methods which rely on optical flow.

Keywords: motion transfer, video-driven image animation, phase-based motion processing

ACM CCS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Imparting gentle motion to objects in an image ascribes a sense of realism to the scene, producing a natural, artistic effect. In this paper, we propose a novel method that uses a sample video to animate objects in an image that are expected to move in a gentle, stochastic and/or oscillatory fashion (see Figure 1). Typically, we find such motion in images of natural objects, for example, in gently rippling water in a lake, flickering candles, smoke.

Past efforts in animating such natural images can be categorized into two classes: (1) formulating motion models for animating target objects and (2) using video input to drive image animation. The first category requires a sound mathematical model for each kind of moving object and for each driving force of motion (e.g. wind) [CGZ*05]. Generalizing such a framework to a large variety of objects and motion types is challenging and requires extensive user input, user-specified parameters and tuning. The second category of techniques provides a promising alternative: the animation process can be directed by simply providing a video example of the desired animation effect. Unlike the first category, there is no need to tune motion settings or devise motion models for each kind of target object.

Our proposed method falls into the second category. The user provides an example video from which motion information is derived and transferred to user-specified similar regions of the input image. The key difference between past approaches for video-driven image animation and ours is in the manner in which motion information is derived from the video input. Previous approaches use image-intensity measures for motion information, such as optical flow [LYT11], average flow field and residues [OAIS09, GMYC12, OAO11]. In contrast, our approach uses phase variations in the example video frames to extract the motion information.

The effectiveness of using phase-based measures over intensity-based measures for random and/or oscillatory motion has been proven for applications like motion magnification [WRDF13] and view interpolation [MWZ*15]. We extend the application of phase-based motion processing techniques to image animation driven by

Figure 1: *Examples of images animated using our algorithm. Regions which move gently, such as trees, water in lake, smoke, candle flame, grass are imparted motion using our algorithm. Image credits (left to right for each row): Makoto Okabe/Wiley Publications, Pierre Leclerc/Shutterstock.com, Pavel L Photo and Video/Shutterstock.com, Active Stock/Shutterstock.com, http://clicktoseeworld.blogspot.com, Makoto Okabe/Wiley Publications.*

video examples and demonstrate its effectiveness over an existing intensity-based approach [OAO11].

Our phase-based motion transfer method is an Eulerian motion processing technique [WRDF13, WRS*12]. Such techniques do not require explicit motion estimation for motion processing tasks. This is in contrast to the Lagrangian motion processing techniques, such as optical flow, which require explicit computation of motion flow. Explicit motion estimation can be computationally expensive and particularly difficult for stochastic motion. Thus, Eulerian approaches give more favourable results over Lagrangian approaches for such kind of motion. In this work, we have adopted an Eulerian approach for motion processing because we focus on stochastic motion that occurs in natural images.

2. Related Work

Video texture synthesis: Synthesizing an animation from an input video footage or a still image has been an extensive area of study in the past, both for computer graphics and computer vision researchers. Many researchers have focused on synthesizing animations using an input video footage or a sequence of images along with user-provided 3D scene information [BSHK04, SSSE00, KSE*03, DCWS03, HDK07, HAA97, BAAR12]. Another active area of research has been the synthesis of time-varying textures. Fluid flow [NKL*07], flow fields [KEBK05] and example textures [WL00] drive such methods. Our work is similar in spirit to such video texture synthesis methods since the end goal for all such techniques is to generate a realistic animation.

Single image animation: Another area of research on generating realistic animations focuses on animating a single image using some prior knowledge. Chuang *et al.* [CGZ*05] propose a semi-automatic approach to animate an image of a natural scene assuming wind as the driving force for motion. There is a fair amount of user interaction involved in marking out the target object with reasonable accuracy and some trials to arrive at satisfactory motion parameters. Wang *et al.* [WZ03] animate an image using a motion model applied to a decomposition of an image over a Gabor or a Fourier basis followed by statistical learning to synthesize animation. Compared to these methods, our approach is simpler and more generalizable since we rely on videos to extract motion information instead of object-specific motion models or user-specified motion parameters. User interaction is limited to marking out regions of interest and a few corresponding strokes.

Due to its flexibility, using video inputs to drive image animation is promising and has spurred many research directions in the past [OAIS09, OAO11, GMYC12]. Okabe *et al.* [OAIS09] animate an image using an example video after matching the average optical flow in the video to the expected average optical flow (specified by the user) in the input image. Intensity variations in the matched video are transferred to the input image to animate it. A later work [OAO11] improves this approach by utilizing a database of video patches and searching for best patches for each patch in the input image. The intensity variations from matched video patches are transferred to the image to animate it. In Section 4, we compare against this work to demonstrate the effectiveness of our approach.

Phase-based motion processing: There have been several other applications of using phase-based measures for analysing small motion in video using other kinds of decompositions like PCA [DAJP11]. Some techniques that use phase-based motion measures perform video motion magnification [WRDF13], view interpolation [MWZ*15], material property estimation [DBC*15] and capturing object properties through a video and synthesizing animation in response to external forces on the object [DCD15]. We expand the application of phase-based motion processing techniques to video-driven image animation. We leverage the ideas developed in [WRDF13, MWZ*15] in our technique for motion transfer.

3. Algorithm

Our goal is to animate a static image, $I(x, y)$, using a similar example video, $V(x, y, t)$, to produce the animated frames $I(x, y, t_1), \ldots, I(x, y, t_n)$. We assume that $V(x, y, t)$ is stabilized or captured using a tripod mounted camera. If the example video has camera motion, then the phase changes due to gentle motion of target objects are masked by the more drastic phase changes introduced by camera shake. However, this kind of motion can be removed by video stabilization algorithms.

There are three stages in our proposed algorithm (Figure 2): user input, video alignment and motion transfer. First, the user marks out the part of the image to be animated, the corresponding region in the example video and optionally provides some sparse corresponding strokes (Section 3.2). We then compute a dense correspondence between these regions of interest (ROIs) and align the region in the video frame to better match the content of the ROI in the input image. Finally, motion information from the aligned video is transferred to the input image by computing the phase shifts in the sub-bands of the example video and adding these to the phase in the corresponding sub-bands of the input image.

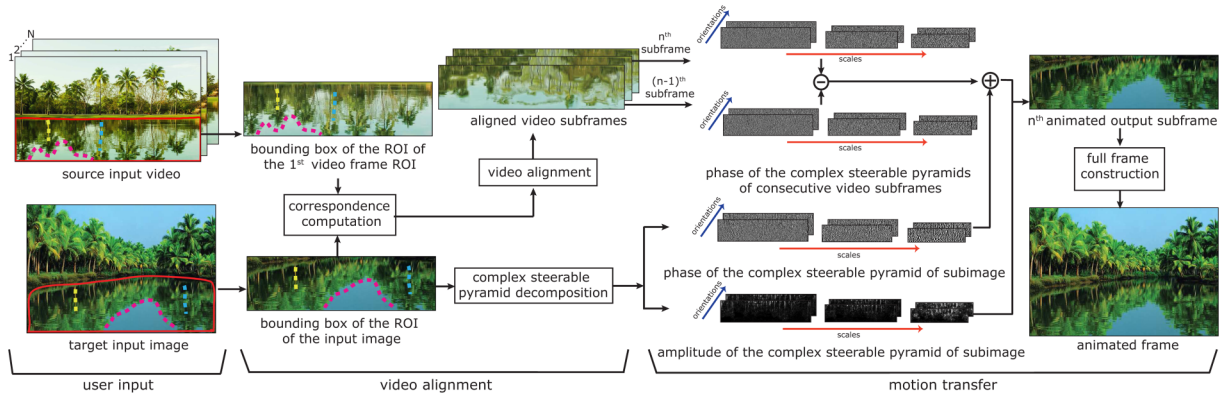


Figure 2: *Overview of our algorithm. The proposed method has three main stages: user input, video alignment and motion transfer. The user supplies the contours of the region of interest (ROI) to be animated in the input image and a corresponding contour around a similar-looking region in the example video (marked out in red in the image and the video to the left). To facilitate the estimation of a correspondence, the user can mark a sparse set of corresponding strokes in the ROIs (shown as dashed lines). The next few steps are performed on sub-frames extracted using the bounding boxes of the ROIs in the image and the video. A correspondence is established between the sub-image and the first video sub-frame. This correspondence is used to align all the subsequent video sub-frames. Motion is transferred using phase changes in the complex steerable pyramid decomposition from the aligned video sub-frames to the input sub-image. Finally, the full animated output frame is reconstructed by combining the resulting animation of the ROI portion of the sub-image with the residual input image that is outside the ROI. Image credits: source input video- pzAxe/Shutterstock.com, target input image- http://clicktoseeworld.blogspot.com.*

We begin the detailed explanation of our method by first building an intuition on how we can use phase for motion transfer.

3.1. Phase-based motion

Our explanation in this section is based on the explanation of previous work done in the area of phase based motion processing [WRDF13, MWZ*15]. Consider a function, $f(x)$ (e.g. see Figure 3a), with the Fourier decomposition:

$$f(x) = \sum_{\omega=-\infty}^{\infty} A_\omega e^{i\omega x}. \tag{1}$$

If $f(x)$ were spatially shifted by an amount Δx, this would result in a phase shift of $\omega\Delta x$ (or a group delay of Δx) for each frequency component ω (in accordance with the Fourier shift theorem [OSB99]),

$$f(x + \Delta x) = \sum_{\omega=-\infty}^{\infty} A_\omega \, e^{i\omega(x+\Delta x)}. \tag{2}$$

Conversely, if we had a new static signal, $g(x)$, with the Fourier decomposition:

$$g(x) = \sum_{\omega=-\infty}^{\infty} B_\omega \, e^{i\omega x}, \tag{3}$$

and we want to shift this signal by the same amount as $f(x)$, we can do so by transferring the phase shift of $\omega\Delta x$ between corresponding frequency components of the two functions. This gives,

$$\sum_{\omega=-\infty}^{\infty} B_\omega \, e^{i\omega(x+\Delta x)} = g(x + \Delta x). \tag{4}$$

Let us generalize and say that the group delays of sinusoids in $f(x)$ are not the same for all frequencies. Furthermore, let us consider the case where this frequency-dependent group delay is a function of time, resulting in $f(x,t)$. We denote this time-varying shift for a frequency component at ω as $d_\omega(t)$. $f(x,t)$ is then given by

$$f(x,t) = \sum_{\omega=-\infty}^{\infty} A_\omega \, e^{i\omega(x + d_\omega(t))}. \tag{5}$$

To apply the local changes of $f(x,t)$ to $g(x)$, we can transfer the phase changes from the frequency components of $f(x)$ to corresponding components of $g(x)$ resulting in $g(x,t)$. This is similar to Equation (3.1) except that the group delays for different frequency components are not alike and are time-varying. Specifically, for a time instant t_n, $g(x,t_n)$ is given by

$$g(x,t_n) = \sum_{\omega=-\infty}^{\infty} B_\omega \, e^{i\omega x \, + \, \omega(d_\omega(t_n) \, - \, d_\omega(t_1))}, \tag{6}$$

where $\omega(d_\omega(t_n) - d_\omega(t_1))$ is the phase difference in $f(x,t)$ at frequency component ω between time t_n and an initial time t_1. For the motion in $g(x,t)$ to appear similar to that in $f(x,t)$, it is important that both $g(x)$ (Figure 3b) and $f(x)$ (Figure 3a) have similar frequency content. If this condition holds, the resultant motion in $g(x,t)$ would appear similar to that in $f(x,t)$, which is the desirable outcome. However, in case of dissimilar content, the frequency components that are common to both $g(x)$ and $f(x)$ would have the expected phase shift (as per Equation 3.1) but $g(x,t)$ may not look plausible.

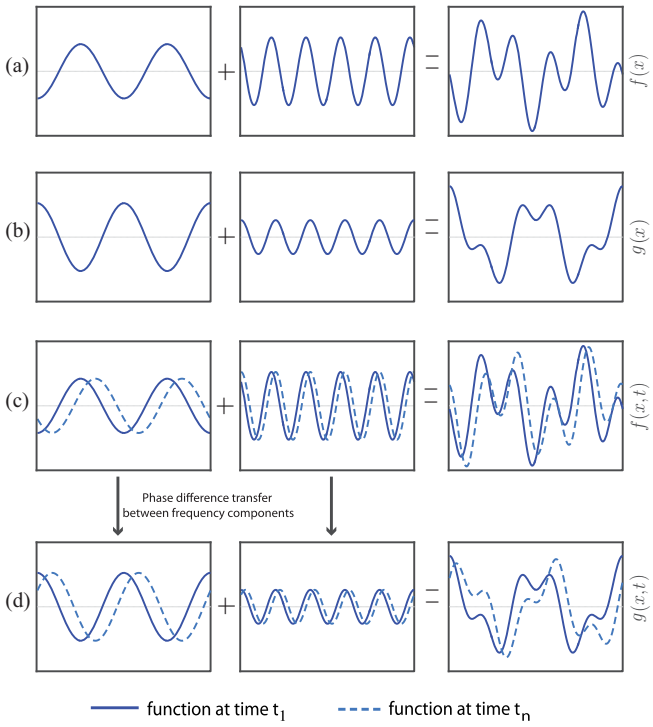


Figure 3: *Phase-based motion transfer in* 1D.*(a)* $f(x)$ *with its constituent frequency components. (b) A similar function* $g(x)$*. (c)* $f(x,t)$*, a spatially shifted version of* $f(x)$*. It is obtained by applying time-varying spatial shifts to* $f(x)$ *and the motion can be resolved into phase shifts of individual frequency components as shown here. (d) The result of transferring phase shifts from* $f(x,t)$ *to the corresponding frequency components of* $g(x)$ *generates* $g(x,t)$*.*

To apply this idea of motion transfer using phase to the case of two-dimensional (2D) functions, we first decompose a 2D function, say $I(x, y)$, into frequency sub-bands. A 2D function can be divided into sub-bands according to spatial frequency, ω, and orientation, θ, by constructing its complex steerable pyramid [SF95]. Except for the low pass and high pass residuals of this decomposition, all sub-bands, $S_{\omega,\theta}(x, y)$, are complex valued and their phase variation can convey meaningful motion to $I(x, y)$.

Just as in the 1D case, if we are given a function moving with time, $V(x, y, t)$, with content similar to the stationary function, $I(x, y)$, we can transfer the phase variations from V to I in the corresponding sub-bands to move I in a plausible manner. The advantage of using a complex steerable pyramid decomposition of functions is that it enables us to capture and transfer phase variations that are localized in space, scale and orientation from $V(x, y, t)$ to $I(x, y)$ [WRDF13, SF95].

3.2. User inputs

Given a static image, $I(x, y)$, and an example video, $V(x, y, t)$, we expect two kinds of inputs from the user:

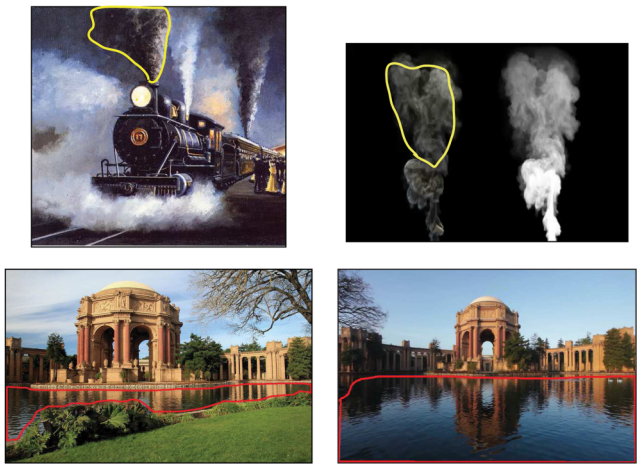


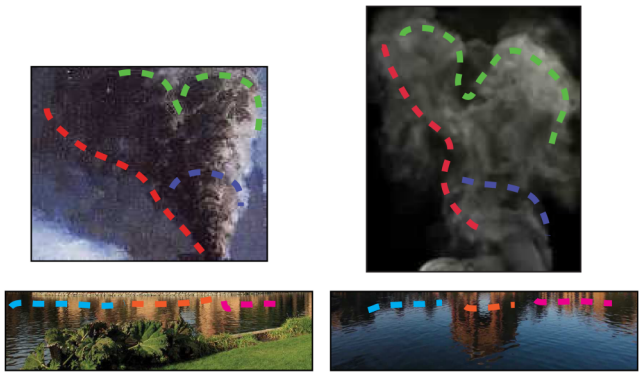
Figure 4: *Contours showing user-marked regions of interest (ROIs). (a) ROI on the input image which is the target image region for animation and (b) ROI on the first frame of the example video which is the source of motion information. Image credits (left to right for each row): Makoto Okabe/Wiley Publications, Arseniy Gutov/Shutterstock.com, Pierre Leclerc/Shutterstock.com, Nickolay Stanev/Shutterstock.com.*

1. *ROI (Figure* 4): The ROI in the image is the region to be animated. The ROI in the video (marked by the user on the first frame) is the region of the video from where the motion information is to be extracted. The ROI marked on the first frame is propagated to all the frames as is. This works because $V(x, y, t)$ is stabilized and our approach is intended for transferring small, stochastic and/or oscillatory motion which may be typically contained within the video ROI.
2. *Matching stroke pairs between the ROIs (Figure* 5): Two to three matching strokes can be optionally marked by the user to identify contours/features in the image and their similar-looking counterparts in the example video frame (Figure 5).

Since computing a steerable pyramid for irregularly shaped ROIs can be inaccurate and not well defined, we use bounding boxes of the ROIs for extracting and transferring motion information. After the animation process is complete, each frame of the output video is generated by combining every frame of the animated ROI(s) of the input image with the rest of the image (see Section 3.4).

3.3. Video alignment

$\widetilde{V}(x, y, t)$ and $\widetilde{I}(x, y)$ are the video and image content in the ROI bounding boxes, respectively. Since the content in $\widetilde{V}(x, y, t)$ is not exactly the same as in $\widetilde{I}(x, y)$, we first establish a dense correspondence between the first video sub-frame, $\widetilde{V}(x, y, t_1)$, and $\widetilde{I}(x, y)$, which for each pixel in $I(x, y)$, identifies a source pixel for motion information from $V(x, y, t)$. Moreover, a correspondence map is useful to handle the cases when the ROIs in $\widetilde{V}(x, y, t)$ and $\widetilde{I}(x, y)$ differ in size, shape or placement within the bounding box and are



(a) User-marked strokes on the image ROI

(b) User-marked strokes on the first video frame ROI

Figure 5: *User-marked matching stroke pairs between the image's and video's ROI. This figure shows the bounding boxes of (a) the image's and (b) the video's ROI. The user marks a set of strokes (typically 3) in the input image's ROI and corresponding strokes in the video's ROI. In this figure (and in subsequent figures), the strokes are colour coded to indicate the correspondence. Image credits (left to right for each row): Makoto Okabe/Wiley Publications, Arseniy Gutov/Shutterstock.com, Pierre Leclerc/Shutterstock.com, Nickolay Stanev/Shutterstock.com.*

not related by a simple image transformation (e.g. the ROIs shown in Figure 4).

We utilize SIFT flow [LYT11] to establish a correspondence between $\widetilde{V}(x, y, t_1)$ and $\widetilde{I}(x, y)$. The SIFT flow algorithm has the desirable property of explicitly enforcing spatial smoothness in its formulation, besides other constraints. We use SIFT flow with the following changes:

1. Instead of the default zero offset initialization of the correspondence map, we initialize SIFT flow with our first guess of the correspondence map. We obtain this initial guess using the sparse corresponding user-marked strokes: correspondence information in between the corresponding strokes is derived by a linear extrapolation of the correspondence values obtained from the user-marked strokes (Figure 5), resulting in a dense correspondence.
2. The original SIFT flow algorithm adopts a multi-scale approach to search for a correspondence map. Belief propagation is performed on the factor graph at each scale to search for a plausible correspondence within a fixed window using the correspondence at the previous coarse scale as the initial belief. Since we already have an initial guess for the correspondence at the finest scale, we apply the SIFT flow algorithm on just the finest scale using this initial guess. We empirically fix the search window size for the optimal correspondence to 20 pixels around the initial correspondence values (see Section 4 for implementation details).

To justify these design choices in the video alignment process, in the supplementary video, we show the final animations generated using three approaches to estimate the correspondence for two examples:

1. Default SIFT flow algorithm with a multi-scale approach and zero offset initialization.
2. Initial correspondence obtained by extrapolating sparse user strokes.
3. Our proposed approach.

It is clear from these results that a simple extrapolation-based correspondence or default SIFT flow may not produce plausible motion in all regions of the image's ROI.

3.4. Motion transfer

Once we have computed our final correspondence map using the method described in previous section, we use it to warp each frame of $\widetilde{V}(x, y, t)$ to align the video content to $\widetilde{I}(x, y)$. Let us denote the aligned video as $\widehat{V}(x, y, t)$. We will transfer motion from $\widehat{V}(x, y, t)$ to $\widetilde{I}(x, y)$.

As mentioned in Section 3.1, phase variations in spatial sub-bands of $\widehat{V}(x, y, t)$ can impart meaningful motion to $\widetilde{I}(x, y)$ by transferring the phase changes to the corresponding sub-bands of $\widetilde{I}(x, y)$. We use the complex steerable pyramid decomposition [SF95, WRDF13] to compute and transfer these phase variations to animate $\widetilde{I}(x, y)$.

Phase computation for the resultant animation: The complex steerable pyramid is computed for each frame of $\widehat{V}(x, y, t)$ and for $\widetilde{I}(x, y)$. Note that this is only computed over the bounding box of an ROI (not the full frame). The phase of the nth animated frame, denoted as $\widetilde{\phi}^n_{\omega,\theta}(x, y)$ for the sub-band at frequency ω and orientation θ, is given by,

$$\widetilde{\phi}^n_{\omega,\theta}(x, y) = \widetilde{\phi}^{n-1}_{\omega,\theta}(x, y) + \Delta\widehat{\Phi}^{n,n-1}_{\omega,\theta}(x, y), \tag{7}$$

where $\Delta\widehat{\Phi}^{n,n-1}_{\omega,\theta}(x, y)$ is the phase difference between the nth and the $(n-1)$th frame of $\widehat{V}(x, y, t)$. In this formulation, $\widetilde{\phi}^1_{\omega,\theta}(x, y)$ is the phase of the input image.

nth output frame reconstruction: Each sub-band, denoted as $\widetilde{S}^n_{\omega,\theta}(x, y)$ for the sub-band at frequency ω and orientation θ, of the nth animated frame is constructed using the phase from Equation (3.4) and the amplitude of $\widetilde{I}(x, y)$, which we denote as $\widetilde{A}_{\omega,\theta}(x, y)$:

$$\widetilde{S}^n_{\omega,\theta}(x, y) = \widetilde{A}_{\omega,\theta}(x, y)e^{\widetilde{\phi}^n_{\omega,\theta}(x,y)}. \tag{8}$$

The final reconstructed frame, $\widetilde{I}^n(x, y)$, is obtained by combining the spatial sub-bands $\widetilde{S}^n_{\omega,\theta}(x, y)$ with the residual frequency components (see Section 3.1) of $\widetilde{I}(x, y)$.

Given $\widetilde{I}^n(x, y)$, we simply compute the full-sized output frame, $I(x, y, t_n)$, by incorporating the original information outside of the image ROI as follows:

$$I(x, y, t_n) = \begin{cases} \widetilde{I}^n(x, y), & \text{if } (x, y) \in \text{ ROI of } I(x, y) \\ I(x, y), & \text{otherwise.} \end{cases} \tag{9}$$

This simple strategy gives plausible animation without edge artefacts (see the supplementary video for results) for gentle, oscillatory and/or stochastic motion as long as the ROIs marked by the user on the input images are reasonably accurate (as shown in Figure 7).

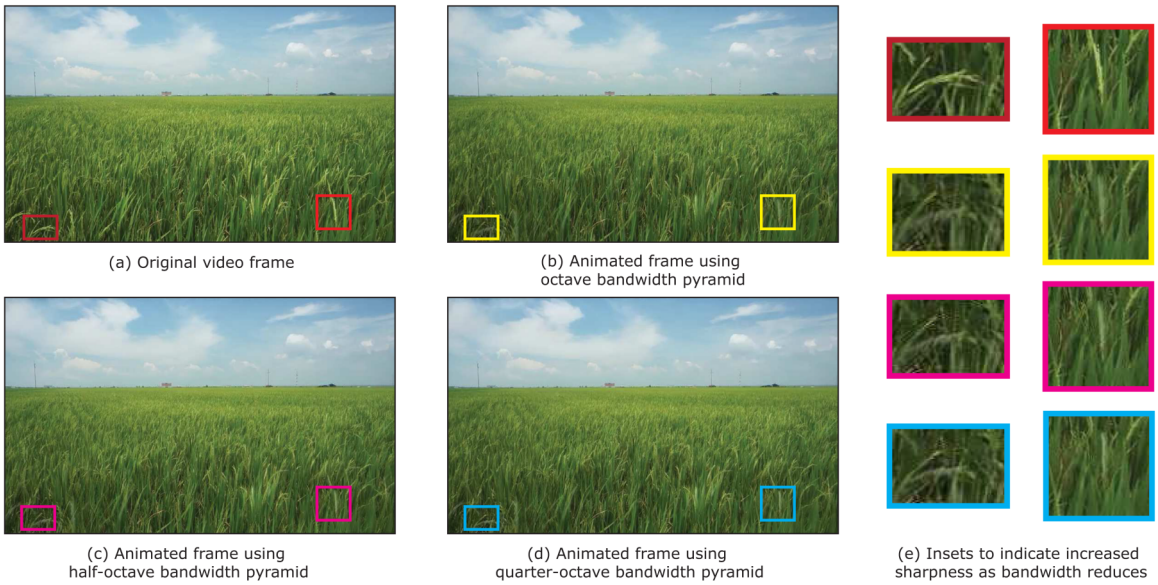


Figure 6: *Effect of changing the filter bandwidth on the animated results. (a) Original video. The animations in (b), (c) and (d) are generated by extracting phase variations from the original video and transferring them to the first frame of the video itself. Thus, the video alignment step of the animation pipeline is unnecessary and we can directly see the effect of phase-based motion transfer without interference from errors in correspondence. The animation in (b) is generated using an octave bandwidth complex steerable pyramid, (c) is generated using a half octave bandwidth complex steerable pyramid and (d) is generated using a quarter octave bandwidth complex steerable pyramid. The insets in (e) zoom into specific sub-regions of these results. As is evident from these insets and also from the supplementary video, reducing the bandwidth increases the sharpness of results, especially in the high-frequency regions. Image credit: Marwadi Bahar/Shutterstock.com*

Choice of steerable filters (Figure 6): The amount of motion the Eulerian motion processing techniques can accurately capture is directly related to the spatial support of the steerable functions used in the complex steerable pyramid decomposition [WRDF13]. When the motion is larger than the spatial support, ringing and blurring artefacts appear. Spatial support of the filters is related to their bandwidth in the frequency domain: a larger bandwidth implies a narrow spatial support and vice versa. A narrow spatial support (wide filter bandwidth) is favourable for the purpose of capturing local motion, but limits the amount of motion that can be captured. We find a good balance of this trade-off by using a default filter bandwidth of a half octave [WRDF13] and allowing the user to reduce the bandwidth to a quarter octave in case the amount of motion is larger. This increases the amount of motion that is supported by our proposed approach, up to a limit. A derivation of the limit to the amount of motion that can be captured by reducing the bandwidth can be found in a previous work on phase-based motion magnification [WRDF13]. Figure 6 and the supplementary video indicate the effect of reducing the filter bandwidth on the resulting animation.

4. Results

Our algorithm enables users to animate static images by transferring motion from example videos using a phase-based measure for motion. We implemented our proposed method in MATLAB on a computer with 8 GB RAM and 2.8 GHz processor. Our unoptimized code took under 2 s (depending on the typical sizes of the ROIs shown in Figure 7) to generate an output frame after the correspondence has been established with the help of the user input. The algorithm consists of few parameters which are constant for the results shown in Figure 7. Mainly, these include the settings for computing a steerable pyramid: filter bandwidth, number of scales and number of orientations. The complex steerable pyramid is constructed with a half octave bandwidth, and eight orientations. The total number of scales (excluding the low pass residue) for the half octave bandwidth pyramid is given by $2(\log_2 s - 2)$, where s is the size of the smaller dimension of the input image's ROI.

Other parameters involved are the settings for the SIFT flow algorithm. We keep the parameters involved in the optimization to their default values provided in the MATLAB demo code for SIFT flow [LYT11] except for the search window size, which is set to 20 pixels at the finest scale based on our experiments. All the computation of Section 3.4 is performed on the luminance channel of the Lab colour space [WRDF13]. This prevents the introduction of colour distortions in the resulting animation, which may arise due to the difference in colour tones of the example video and the input image.

Figure 7 depicts the image and video inputs that we used to test our algorithm. The user supplies the example videos for each unique kind of motion to be achieved in the image. For example, in Figure 7(g), two example videos are supplied, one for animating the waterfall and the other for animating the river since the motion of

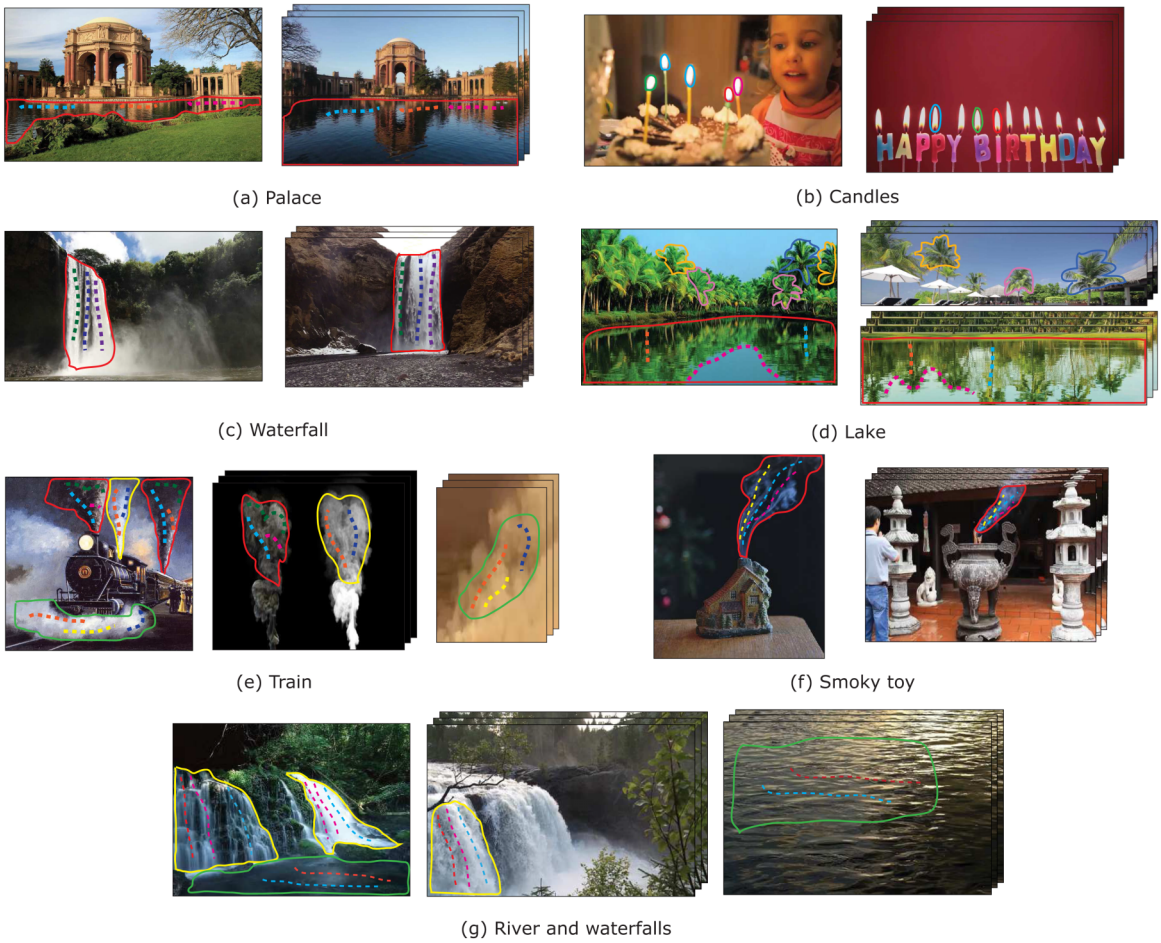
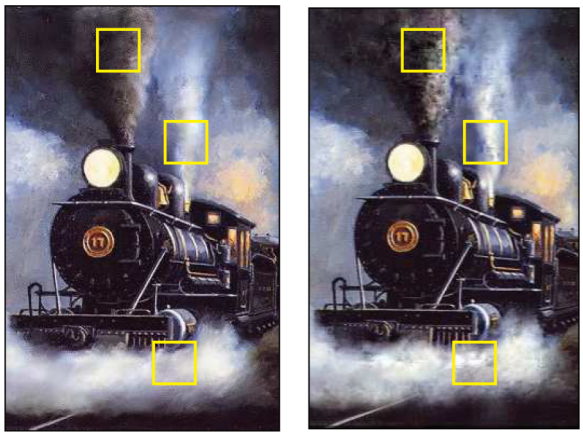


Figure 7: *Inputs to our algorithm. For each sub-figure, the input image is shown on the left and the example video(s) used for animation are to the right. ROIs marked by the user are shown in solid lines and colour coded to show the correspondence between the image and the video. Similarly, user strokes within the ROIs (if needed) are marked with dashed lines and colour coded for correspondence. For small objects, like candles and trees, user strokes were not necessary. As can be seen from our results, we support animation of a variety of objects like candles, waterfall, rippling lake, trees swaying in wind, smoke and stream which exhibit small motion in a stochastic or oscillatory manner. Please view the supplementary video for the resulting animations. Image credits (result name - image credit, example video(s) credit): Palace - Pierre Leclerc/Shutterstock.com, Nickolay Stanev/Shutterstock.com; Candles - Pavel L Photo and Video/Shutterstock.com, wavebreakmedia/Shutterstock.com; Waterfall - Active Stock/Shutterstock.com, Narongsak Nagadhana/Shutterstock.com; Lake - http://clicktoseeworld.blogspot.com, Wolfgang Amri/Shutterstock.com, pzAxe/Shutterstock.com; Train: Makoto Okabe/Wiley Publications, Arseniy Gutov/Shutterstock.com, Realstock/Shutterstock.com; Smoky toy: Makoto Okabe/Wiley Publications, Pruda/Shutterstock.com; River and waterfalls: Makoto Okabe/Wiley Publications, JKlingebiel/Shutterstock.com, Mino Surkala/Shutterstock.com.*

both these elements is quite different. User input in the form of ROI borders and matching strokes is required as discussed in Section 3.2. Please view the supplementary video for the resulting animation.

We visually compare (Figure 8) the performance of our algorithm with the work on animating images using video examples by Okabe *et al.* [OAO11]. This is a data-driven method to animate fluid regions in an image using a database of patches from videos of fluids. For each patch in the input image, a suitable patch is retrieved from this database. The selection of a suitable patch depends on three criteria: first, the visual similarity between the image and the video patch; second, the spatial agreement of the selected video patch with selected neighbouring video patches; and third, the similarity of the optical flow of the candidate video patch with the desired optical flow in the input image patch. The desired optical flow is obtained from the user-specified motion field which is comprised of an orientation and a speed map for the desired animation. The pixel intensity differences between the frames of the selected video patches are suitably transferred to the image patches to animate the image.



(a) A frame of the animated result using our approach

(b) A frame of the animated result by Okabe et al. 2011

Figure 8: *Comparison with Okabe et al. 2011 [OAO11]: (a) exemplary frame of the animated video generated using our algorithm; (b) shows the frame obtained from the animation result of [OAO11]. Since their method relies on intensity-based measures for motion and video patches, the animated results show blocking and colour artefacts. Our phase-based motion measure is free of such artefacts. The yellow boxes in the figure highlight some regions for comparison. Please refer to the supplementary video for an animated comparison. Image credit: Makoto Okabe/Wiley Publications.*

(a) Original video

(b) A frame of the animated result using our approach

Figure 9: *Limitation of our approach. The animation in (b) is created by transferring phase variations from the source video shown in (a) (just as we did for Figure 6). The dramatic flicker of fire is not entirely captured by the phase changes in our method. This shortcoming manifests as blurring and ringing in the resulting animation. Please refer to the supplementary video to view the full animation. In particular, we would like to draw the viewers' attention to the yellow regions marked in (b).*

Compared to such an approach, our proposed method is simpler not only in terms of user interaction, but also in extracting motion information from video. A key advantage of our approach comes from the use of the phase-based measure for motion, which is less prone to artefacts as compared to the pixel intensity difference-based measure used by Okabe *et al.* [OAO11]. Besides, since Okabe *et al.* utilize video patches instead of a single video, certain blocking artefacts are visible when there is poor agreement among the neighbouring selected example video patches. An instance where such artefacts are visible is shown in Figure 8(b) along with the favourable performance of our algorithm in such cases in Figure 8(a). Please refer to the supplementary video for resulting animations. Our results for the comparison were generated using the input images shown in Figure 7(e), (f) and (g), supplied to us by the authors of the paper. The example videos and user inputs used to generate the results are also shown in Figure 7(e), (f) and (g) and the supplementary video.

Limitations: Due to the limited spatial support of the steerable filter used in the pyramid decomposition of the image and the video frames (Section 3.4), our approach is only suitable for transferring gentle, stochastic and/or oscillatory motion. While such motion is prevalent in a variety of objects (Figure 7), animating objects with drastic motion can result in ringing and blurring artefacts (Figure 9). Such limitations might be overcome by complementing the Eulerian techniques with Lagrangian approaches for large motions. This is a direction for future research in this area.

5. Conclusions and Future Work

We have proposed a Eulerian method for animating static images using one or more example videos as inputs. In contrast to the Lagrangian approaches that rely on explicitly estimating motion from the input video, such as optical flow, our phase-based Eulerian method creates pleasing animations on scenes where optical flow is not robust, such as objects with subtle and stochastic motion. We have demonstrated the applicability of the method for animating different types of objects, such as rippling water, swaying grass, flickering candles, smoke and waterfall.

Furthermore, our technique does not require complex mathematical motion modeling and does not require the user to provide motion parameters; the motion information is extracted from the provided input video. This is a promising alternative to the use of complex motion models for image animation.

Acknowledgements

We would like to thank Makoto Okabe for providing some of the source images for animation. We thank Timo Ahonen and other colleagues at Nokia for helpful advice and Abhishek Badki for his help with putting together the results. This work was funded in part by National Science Foundation grant IIS-1321168.

References

[BAAR12] Bai J., Agarwala A., Agrawala M., Ramamoorthi R.: Selectively de-animating video. *ACM Transactions on Graphics 31*, 4 (July 2012), 66:1–66:10.

[BSHK04] Bhat K. S., Seitz S. M., Hodgins J. K., Khosla P. K.: Flow-based video synthesis and editing. In ACM SIGGRAPH 2004 Papers (New York, NY, USA, 2004), SIGGRAPH '04, ACM, pp. 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 ACM SIGGRAPH 2005 Papers (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 853–860.

[DAJP11] Dixon M., Abrams A., Jacobs N., Pless R.: On analyzing video with very small motions. In 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Providence, RI, June 2011), IEEE pp. 1–8.

[DBC*15] Davis A., Bouman K. L., Chen J. G., Rubinstein M., Durand F., Freeman W. T.: Visual vibrometry: Estimating material properties from small motion in video. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (Boston, MA, June 2015), IEEE.

[DCD15] Davis A., Chen J. G., Durand F.: Image-space modal bases for plausible manipulation of objects in video. *ACM Transactions on Graphics 34*, 6 (October 2015), 239:1–239:7.

[DCWS03] Doretto G., Chiuso A., Wu Y. N., Soatto S.: Dynamic textures. *International Journal of Computer Vision 51*, 2 (February 2003), 91–109.

[GMYC12] Gui Y., Ma L., Yin C., Chen Z.: Preserving global features of fluid animation from a single image using video examples. *Journal of Zhejiang University: Science C 13*, 7 (2012), 510–519.

[HAA97] Horry Y., Anjyo K.-I., Arai K.: Tour into the picture: Using a spidery mesh interface to make animation from a single image. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 225–232.

[HDK07] Hornung A., Dekkers E., Kobbelt L.: Character animation from 2D pictures and 3D motion data. *ACM Transactions on Graphics 26*, 1 (January 2007).

[KEBK05] Kwatra V., Essa I., Bobick A., Kwatra N.: Texture optimization for example-based synthesis. *ACM Transactions on Graphics 24*, 3 (July 2005), 795–802.

[KSE*03] Kwatra V., Schödl A., Essa I., Turk G., Bobick A.: Graphcut textures: Image and video synthesis using graph cuts. In ACM SIGGRAPH 2003 Papers (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 277–286.

[LYT11] Liu C., Yuen J., Torralba A.: SIFT flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 5 (2011), 978–994.

[MWZ*15] Meyer S., Wang O., Zimmer H., Grosse M., Sorkine-Hornung A.: Phase-based frame interpolation for video. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Boston, MA, June 2015), IEEE, pp. 1410–1418.

[NKL*07] Narain R., Kwatra V., Lee H.-P., Kim T., Carlson M., Lin M. C.: Feature-guided dynamic texture synthesis on continuous flows. In Proceedings of the 18th Eurographics Conference on Rendering Techniques (Aire-la-Ville, Switzerland, Switzerland, 2007), EGSR'07, Eurographics Association, pp. 361–370.

[OAIS09] Okabe M., Anjyo K., Igarashi T., Seidel H.-P.: Animating pictures of fluid using video examples. Computer Graphics Forum *28*, 2 (2009), 677–686.

[OAO11] Okabe M., Anjyo K., Onai R.: Creating fluid fnimation from a single image using video database. *Computer Graphics Forum 30*, 7 (2011), 1973–1982.

[OSB99] Oppenheim A. V., Schafer R. W., Buck J. R.: *Discrete-time Signal Processing (2nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[SF95] Simoncelli E., Freeman W.: The steerable pyramid: A flexible architecture for multi-scale derivative computation. In Proceedings of International Conference on Image Processing, 1995. (Washington, DC, October 1995), Vol. *3*, pp. 444–447.

[SSSE00] Schödl A., Szeliski R., Salesin D. H., Essa I.: Video textures. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 489–498.

[WL00] Wei L.-Y., Levoy M.: Fast texture synthesis using tree-structured vector quantization. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co, pp. 479–488.

[WRDF13] Wadhwa N., Rubinstein M., Durand F., Freeman W. T.: Phase-based video motion processing. *ACM Transactions on Graphics 32*, 4 (July 2013), 80:1–80:10.

[WRS*12] Wu H.-Y., Rubinstein M., Shih E., Guttag J., Durand F., Freeman W.: Eulerian video magnification for revealing subtle changes in the world. *ACM Transactions on Graphics 31*, 4 (July 2012), 65:1–65:8.

[WZ03] Wang Y., Zhu S.-C.: Modeling textured motion: particle, wave and sketch. In Proceedings of Ninth IEEE International Conference on Computer Vision, 2003. (Nice, France, October 2003), Vol. *1*, pp. 213–220.

Supporting Information

Additional Supporting Information may be found in the online version of this article at the publisher's web site:

Video S1.