Animating Pictures with Stochastic Motion Textures

Yung-Yu Chuang[1,3] Dan B Goldman[1] Ke Colin Zheng[1] Brian Curless[1] David H. Salesin[1,2] Richard Szeliski[2]

[1]University of Washington [2]Microsoft Research [3]National Taiwan University

(a) Japanese Temple (b) Harbor (c) Boat Studio (d) Argenteuil (e) Sunflowers

Figure 1 Sample input images we animate using our technique. The first two pictures are photographs of a Japanese Temple (a) and a harbor (b). The paintings shown in (c) and (d) are Claude Monet's *The Boat Studio* and *The Bridge at Argenteuil*. We also apply our method to Van Gogh's *Sunflower* (e) to animate the flowers. (The last three paintings are courtesy of WebMuseum, http://www.ibiblio.org/wm/.)

Abstract

In this paper, we explore the problem of enhancing still pictures with subtly animated motions. We limit our domain to scenes containing passive elements that respond to natural forces in some fashion. We use a semi-automatic approach, in which a human user segments the scene into a series of layers to be individually animated. Then, a "stochastic motion texture" is automatically synthesized using a spectral method, i.e., the inverse Fourier transform of a filtered noise spectrum. The motion texture is a time-varying 2D displacement map, which is applied to each layer. The resulting warped layers are then recomposited to form the animated frames. The result is a looping video texture created from a single still image, which has the advantages of being more controllable and of generally higher image quality and resolution than a video texture created from a video source. We demonstrate the technique on a variety of photographs and paintings.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.4.9 [Image Processing and Computer Vision]: Applications

Keywords: Animation, image-based animation, image-based rendering, natural phenomena, physical simulation, video texture

1 Introduction

When we view a photograph or painting, we perceive much more than the static picture before us. We supplement that image with our life experiences: given a picture of a tree, we imagine it swaying; given a picture of a pond, we imagine it rippling. In effect, we bring to bear a strong set of "priors," and these priors enrich our perception.

http://grail.cs.washington.edu/projects/StochasticMotionTextures/

In this paper, we explore how a set of explicitly encoded priors might be used to animate still images on a computer. The *fully automatic* animation of *arbitrary* scenes is, of course, a monumental challenge. In order to make progress, we make the problem easier in two ways.

First, we use a semi-automatic, user-assisted approach. In particular, a user segments the scene into a set of animatable layers and assigns certain parameters to each one. Second, we limit our scope to scenes containing passive elements that respond to natural forces in some fashion. We explore a range of passive elements including plants and trees, water, floating objects such as boats, and clouds. The motion of each of these objects is driven by a single natural force, namely, the wind. Although this set of objects and motions may seem limited, they occur in a large variety of pictures and paintings, as shown in Figure 1.

We have found that all of these elements can be animated using a unified approach. First, we segment the picture into a set of user-specified layers using Bayesian matting [Chuang et al. 2001]. As each layer is removed from the picture, "inpainting" is used to fill in the resulting hole. Next, the user annotates one or more layers with a *motion armature*, a line segment which approximates the structure of a layer. Using these constraints, we synthesize a *stochastic motion texture* using spectral methods [Stam 1995]. Spectral methods work by generating a random noise spectrum in the frequency domain, applying a physically based spectrum filter to that noise, and computing an inverse Fourier transform to create the stochastic motion texture. This motion texture is a time-varying 2D displacement map, which is applied to the pixels in the layer. Finally, the warped layers are recomposited to form the animated picture for each frame.

The resulting moving picture can be thought of as a kind of video texture [Schödl et al. 2000]—although, in this case, a video texture created from a single static image rather than from a video source. Thus, these results have potential application wherever video textures do, i.e., in place of still images on Web sites, as screen savers or desktop "wallpapers," or in presentations and vacation slide shows.

In addition, there are several advantages to creating video textures from a static image rather than from a video source. First, because they are created synthetically, they allow greater creative control in their appearance. For example, the wind direction and amplitude

can be tuned for a particular desired effect. Second, consumer-grade digital still cameras generally provide much higher image quality and greater resolution than their video camera counterparts. These advantages allow animated stills to be used in new situations such as animated matte paintings for special effects. Furthermore, they can be applied to sources that exist only in a static form such as paintings and historic photographs.

For the most part, the algorithms we describe in this paper are applications of techniques from a variety of disparate sources such as image matting and inpainting, and physically based animation of natural phenomena. We show how these techniques can be combined, seamlessly and synergistically into an easy-to-use system for animating still images. Thus, our major contributions are in the formulation of the overall problem, including the recognition that an interesting class of phenomena can all be animated attractively via a single wind source using simple controls; the marshalling of a variety of techniques, most notably stochastic motion textures, to solving this problem; the design of a user interface that allows novice users to animate pictures with little or no training; and lastly, a proof of the viability and quality of applying image warping approaches to synthesizing appealing animated pictures.

1.1 Related work

Our goal is to synthesize a stochastic video from a single image. Hence, our work is similar in spirit to the work on video textures and dynamic textures [Szummer and Picard 1996; Schödl et al. 2000; Wei and Levoy 2000; Soatto et al. 2001; Wang and Zhu 2003]. Like our work, video textures focus on "quasi-periodic" scenes. However, the inputs to video texture algorithms are short videos that can be analyzed to mimic the appearance and dynamics of the scene. In contrast, the input to our work is only a single image.

Our work is, in spirit, similar to the "Tour Into the Picture" system developed by Horry *et al.* [1997]. Their system allows users to map a 2D image onto a simple 3D box scene based on some interactively selected perspective viewing parameters such as vanishing points. This approach allows users to interactively navigate into a picture. Criminisi *et al.* [2000] propose an automated technique that can produce similar effects in a geometrically correct way. More recently, Oh *et al.* [2001] developed an image-based depth editing system capable of augmenting a photograph with a more complicated depth field to synthesize more realistic effects. In our work, instead of synthesizing a depth field to change the viewpoint, we add motion fields to make the scene change over time.

For certain classes of motions, our system requires the user to specify a motion armature for a layer, and then performs physically-based simulation on the armature to synthesize a motion field. It is therefore similar to the method of Litwinowicz and Williams [1994], which uses keyframe line drawings to deform images to create 2D animations. Their system is quite useful for traditional 2D animation. However, their technique is not suitable for modeling the natural phenomena we target because such motions are difficult to keyframe. Also, they use a smooth scattered data interpolation to synthesize a motion field without any physical dynamics.

Our work is also related to the *object-based image editing* system proposed by Barrett and Cheney [2002], namely, *object selection*, *matte extraction*, and *hole filling*. Indeed, Barrett *et al.* have also demonstrated how to generate a video from a single image by editing and interpolating keyframes. Like Litwinowicz's system, the focus is on key-framed rather than stochastic (temporal texture-like) motions.

Freeman *et al.* [1991] previously attempted to create the illusion of motion in a static image in their "Motion without movement" paper. They apply quadrature pairs of oriented filters to vary the local phase in an image to give the illusion of motion. While the motion is quite compelling, the band-pass filtered images do not look photorealistic.

Even earlier, at the turn of the 20th century, people painted outdoor scenes on pieces of masked vellum paper and used series of sequentially timed lights to create the illusion of descending waterfalls [Hathaway et al. 2003]. People still make this kind of device, which is often called a *kinetic waterfall*. Another example of a simple animated picture is the popular Java program, *Lake applet*, which takes a single image and perturbs the image with a set of simple ripples [Griffiths 1997]. Though visually pleasing, these results often do not look realistic because of their lack of physical properties.

Working on an inverse problem to ours, Sun *et al.* [2003] propose a *video-input driven animation* (VIDA) system to extract physical parameters such as wind speed from real video footage. They then use these parameters to drive the physical simulation of synthetic objects to integrate them consistently with the source video. They estimate physical parameters from observed displacements; we synthesize displacements using a physical simulation based on user-specified parameters. They target a similar set of natural phenomena to those we study: plants, waves, and boats, which can all be explained as *harmonic oscillations*.

To simulate dynamics, we use physically-based simulation techniques previously developed in computer graphics for modeling natural phenomena. For waves, we use the Fourier wave model to synthesize a time-varying height field. Mastin *et al.* [1987] were the first to introduce statistical frequency-domain wave models from oceanography into computer graphics. In a similar way, we synthesize stochastic wind fields [Shinya and Fournier 1992; Stam and Fiume 1993] by applying a different spectrum filter. When applying the wind field to trees, since the force is oscillatory in nature, the corresponding motions are also periodic and can be solved more robustly and efficiently in the frequency domain [Stam 1997; Shinya et al. 1998].

Aoki *et al.* [1999] coupled physically-based animations of plants with image morphing techniques as an efficient alternative to expensive physically-based plant simulation and synthesis. However, they only demonstrate their concept on synthetic images. In our work, we target real pictures and use our approach as a way to synthesize video textures for stochastic scenes.

Our system requires users to segment an image into *layers*. To support seamless composites, a soft alpha matte for each layer is required. We use recently proposed interactive image matting algorithms to extract alpha mattes from the input image [Ruzon and Tomasi 2000; Chuang et al. 2001]. To fill in holes left behind after removing each layer, we use an *inpainting* algorithm [Bertalmio et al. 2000; Criminisi et al. 2003; Jia and Tang 2003; Drori et al. 2003].

1.2 Overview

We begin with a system overview that describes the basic flow of our system (Section 2). We then address our most important subproblem, namely synthesizing a stochastic motion texture (Section 3). Finally, we discuss our results (Section 4) and end with conclusions and ideas for future work.

2 System overview

Given a single image, how can we generate a continuously moving animation quickly and easily? One possibility is to use a keyframe-based approach, as did Litwinowicz and Williams [1994]. However, such an approach is problematic for naïve users: specifying the motions is complex, and achieving any kind of movement resembling physical realism is quite difficult. Another straightforward approach is to use compositions of sinusoids to create oscillatory motions

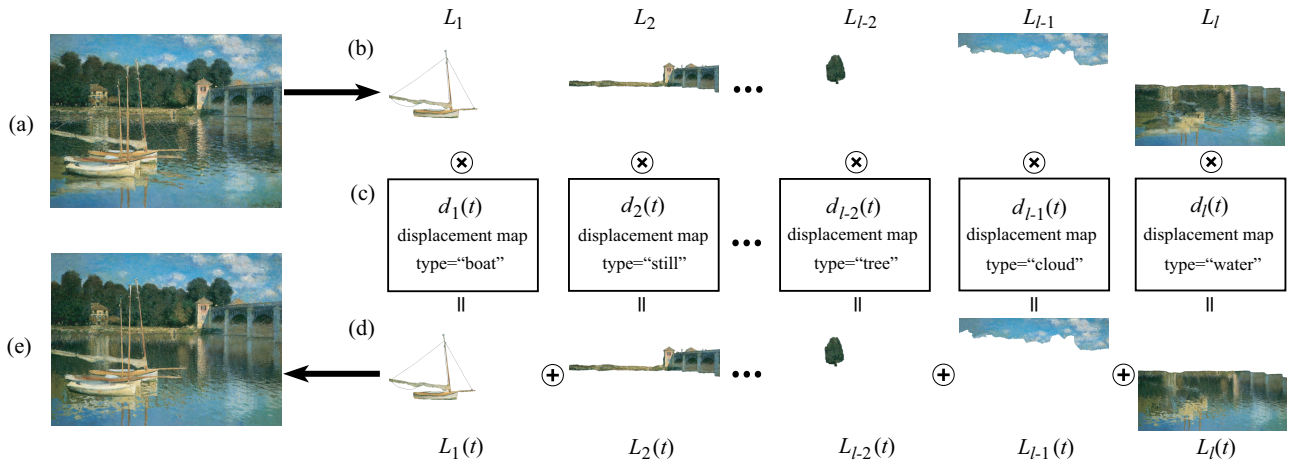

Figure 2 Overview of our system. The input still image (a) is manually segmented into several layers (b). Each layer L_i is then animated with a different stochastic motion texture $d_i(t)$ (c). Finally, the animated layers $L_i(t)$ (d) are composited back together to produce the final animation $I(t)$ (e).

[Griffiths 1997], but the resulting effect may not maintain a viewer's interest over more than a short period of time, on account of its periodicity and predictability.

The approach we ultimately settled upon — which has the advantages of being quite simple for users to specify, and of creating interesting, complex, and plausibly realistic motion — is to break the image up into several layers and to then synthesize a different *motion texture*[1] for each layer. A motion texture is essentially a *time-varying displacement map* defined by a motion type, a set of motion parameters, and in some cases a motion armature. This displacement map $d(p,t)$ is a function of pixel coordinates p and time t. Applying it directly to an image layer L results in a forward warped image layer L' such that

$$L'(p + d(p,t)) = L(p) \tag{1}$$

However, since forward mapping is fraught with problems such as aliasing and holes, we actually use inverse warping, defined as

$$L'(p) = L(p + d'(p,t)) \tag{2}$$

We denote this operation as $L' = L \otimes d'$.

We could compute the inverse displacement map d' from d using the two-pass method suggested by Shade *et al.* [1998]. Instead, since our motion fields are all very smooth, we simply dilate them by the extent of the largest possible motion and reverse their sign.

With this notation in place, we can now describe the basic workflow of our system (Figure 2), which consists of three steps: *layering and matting*, *motion specification and editing*, and finally *rendering*.

Layering and matting. The first step, *layering*, is to segment the input image I into layers so that, within each layer, the same motion texture can be applied. For example, for the painting in Figure 2(a), we have the following layers: one for each of the water, sky, bridge and shore; one for each of the three boats; and one for each of the eleven trees in the background (Figure 2(b)). To accomplish this, we use an interactive object selection tool such as a painting tool or intelligent scissors [Mortensen and Barrett 1995]. The tool is used to specify a *trimap* for a layer; we then apply Bayesian matting to extract the color image and a soft alpha matte for that layer [Chuang et al. 2001].

Because some layers will be moving, occluded parts of the background might become visible. Hence, after extracting a layer, we use an enhanced inpainting algorithm to fill the hole in the background behind the foreground layer. We use an example-based inpainting algorithm based on the work of Criminisi *et al.* [2003] because of its simplicity and its capacity to handle both linear structures and textured regions.

Note that the inpainting algorithm does not have to be perfect since only pixels near the boundary of the hole are likely to become visible. We can therefore accelerate the inpainting algorithm by considering only nearby pixels in the search for similar patches. This shortcut may sacrifice some quality, so in cases where the automatic inpainting algorithm produces poor results, we provide a touch-up interface with which a user can select regions to be repainted. The automatic algorithm is then reapplied to these smaller regions using a larger search radius. We have found that most significant inpainting artifacts can be removed after only one or two such brushstrokes. Although this may seem less efficient than a fully automatic algorithm, we have found that exploiting the human eye in this simple fashion can produce superior results in less than half the time of the fully automatic algorithm. Note that if a layer exhibits large motions (such as a wildly swinging branch), artifacts deep inside the inpainted regions behind that layer may be revealed. In practice, these artifacts may not be objectionable, as the motion tends to draw attention away from them. When they are objectionable, the user has the option of improving the inpainting results.

After the background image has been inpainted, we work on this image to extract the next layer. We repeat this process from the closest layer to the furthest layer to generate the desired number of layers. Each layer L_i contains a color image C_i, a matte α_i, and a compositing order z_i. The compositing order is presently specified by hand, but could in principle be automatically assigned with the order in which the layers are extracted.

Motion specification and editing. The second component of our system lets us *specify* and *edit* the motion texture for each layer. Currently, we provide the following motion types: *trees* (swaying), *water* (rippling), *boats* (bobbing), *clouds* (translation), and *still* (no motion). For each motion type, the user can tune the motion parameters and specify a motion armature, where applicable. We describe the motion parameters and armatures in more detail for each motion type in Section 3.

[1] We use the terms *motion texture* and *stochastic motion texture* interchangeably in this paper. The term *motion texture* was also used by Li et. al [2002] to refer to a linear dynamic system learned from motion capture data.

Since all of the motions we currently support are driven by the wind, the user controls a single wind speed and direction, which is shared by all the layers. This allows all the layers to respond to the wind consistently. Our motion synthesis algorithm is fast enough to animate a half-dozen layers in real-time. Hence, the system can provide instant visual feedback to changes in motion parameters, which makes motion editing easier. Each layer L_i has its own motion texture, d_i, as shown in Figure 2(c).

Rendering. During the *rendering* process, for each time instance t and layer L_i, a displacement map $d_i(t)$ is synthesized. (Here, we have dropped the dependencies of L_i and d_i on p for notational conciseness.) This displacement map is then applied to C_i and α_i to obtain $L_i(t) = L_i(0) \otimes d_i(t)$ (Figure 2(d)). Notice that the displacement is evaluated as an absolute displacement of the input image $I(0)$ rather than a relative displacement of the previous image $I(t-1)$. In this way, repeated resampling and numerical error accumulation are avoided.

Finally, all the warped layers are composited together from back to front to synthesize the frame at time t, $I(t) = L_1(t) \oplus L_2(t) \oplus \ldots \oplus L_l(t)$, where $z_1 \geq z_2 \cdots \geq z_l$ and $\oplus$ is the standard *over* operator [Porter and Duff 1984] (Figure 2(e)).

3 Stochastic motion textures

In this section, we describe our approach to synthesizing the stochastic motion textures that drive the animated image. We first describe the basic principles on which our system is based (Section 3.1). We then describe the details of each motion type, i.e., trees (Section 3.2), water (Section 3.3), bobbing boats (Section 3.4), and clouds (Section 3.5).

3.1 Stochastic modeling of natural phenomena

Many natural motions can be viewed as harmonic oscillations [Sun et al. 2003], and, indeed, hand-crafted superpositions of a small number of sinusoids have often been used to approximate natural phenomena for computer graphics. However, this simple approach has some limitations, as we discovered after experimenting with this idea. First of all, it is tedious to tune the parameters to produce the desired effects. Second, it is hard to create motions for each layer that are consistent with one another since they lack a physical basis. Lastly, the resulting motions do not look natural since they are strictly periodic — irregularity actually plays a central role in modeling natural phenomena.

One way to add randomness is to introduce a noise field. Introducing this noise directly into the temporal or spatial domain often leads to erratic and unrealistic simulations of natural phenomena. Instead, we simulate noise in the frequency domain, and then sculpt the spectral characteristics to match the behaviors of real systems that have intrinsic periodicities and frequency responses. Specific spectrum filters need to be applied to model specific phenomena, leading to so-called *spectral methods* [Stam 1995].

The spectral method for synthesizing a stochastic field has three steps: (1) generate a complex Gaussian random field in the frequency domain, (2) apply a domain-specific spectrum filter, and (3) compute the inverse Fourier transform to synthesize a stochastic field in the time or frequency domain. A nice property of this method is that the synthesized stochastic field can be tiled seamlessly. Hence, we only need to synthesize a patch of reasonable size and tile it to produce a much larger stochastic signal. This tiling approach works reasonably well if the size of the patch is large enough to avoid objectionable repetition. Furthermore, each layer can use a patch of a different size, which obscures any repetitive motion that may remain in individual layers.

To realistically model natural phenomena, the filter should be learned from real-world data. For the phenomena we simulate, plants and waves, such experimental data and statistics are available from other fields, e.g., structural engineering and oceanography, and have already been used by the graphics community to create synthetic imagery [Shinya and Fournier 1992; Stam and Fiume 1993; Mastin et al. 1987]. After experimenting with several different variants published in both the computer graphics and simulation literature, we selected the following set of techniques to synthesize stochastic motion textures that are both realistic and easy to control.

3.2 Plants and trees

The branches and trunks of trees and plants can be modeled as physical systems with mass, damping, and stiffness properties. The driving function that causes branches to sway is typically wind [Stam 1997]. Our goal is to model the spectral filtering due to the dynamics of the branches applied to the spectrum of the driving wind force.

To model the physics of branches, we take the simplified view introduced by Sun *et al.* [2003]. In particular, the motion of each branch is constrained by a motion armature; a 2D line segment parameterized by u, which ranges from 0 to 1. This line segment is drawn by the user for each layer. Note that, to model a correct mechanical structure, the line segment may need to extend outside the image. Displacements of the tip of the branch $d_{\text{tip}}(t)$ are taken to be perpendicular to the line segment. Modal analysis indicates that the displacement perpendicular to the line for other points along the branch can be simplified to the form:

$$d(u,t) = \left[\frac{1}{3}u^4 - \frac{4}{3}u^3 + 2u^2\right] d_{\text{tip}}(t) \tag{3}$$

We approximate the (scalar) displacement of the tip in the direction of the projected wind force as a damped harmonic oscillator:

$$\ddot{d}_{\text{tip}}(t) + \gamma \dot{d}_{\text{tip}}(t) + 4\pi^2 f_o^2 d_{\text{tip}}(t) = w(t)/m \tag{4}$$

where m is the mass of the branch, $f_o = k/m$ is the natural frequency of the system, and $\gamma = c/m$ is the velocity damping term [Sun et al. 2003]. These parameters have a more intuitive meaning than the damping (c) and stiffness (k) terms found in more traditional formulations. The driving force $w(t)$ is derived from the wind force incident on the branch, as detailed below.

Taking the temporal Fourier transform $\mathcal{F}\{\}$ of equation (4) and noting that $\mathcal{F}\{\dot{d}_{\text{tip}}(t)\} = i2\pi f \mathcal{F}\{d_{\text{tip}}(t)\}$, we arrive at

$$-4\pi^2 f^2 D_{\text{tip}}(f) + i2\pi\gamma f D_{\text{tip}}(f) + 4\pi^2 f_o^2 D_{\text{tip}}(f) = \frac{W(f)}{m} \tag{5}$$

where $i = \sqrt{-1}$ and $D_{\text{tip}}(f)$ and $W(f)$ are the Fourier transforms of $d_{\text{tip}}(t)$ and $w(t)$, respectively. Solving for $D_{\text{tip}}(f)$ and expressing the result in complex exponential notation gives

$$D_{\text{tip}}(f) = \frac{W(f) e^{i2\pi\theta}}{2\pi m \left\{\left[2\pi\left(f^2 - f_o^2\right)\right]^2 + \gamma^2 f^2\right\}^{1/2}} \tag{6}$$

where $W(f)$ is the Fourier transform of the driving wind force, a function of frequency f, as defined in equations (8) and (9) below. The phase shift θ is given by

$$\tan\theta = \frac{\gamma f}{2\pi\left(f^2 - f_o^2\right)} \tag{7}$$

Next, we model the forcing spectrum for wind. An empirical model made from experimental measurements [Simiu and Scanlan 1986, p. 55] indicates that the temporal power spectrum of the wind velocity at a point takes the following form:

$$P_V(f) \sim \frac{v_{\text{mean}}}{\left(1 + \kappa f / v_{\text{mean}}\right)^{5/3}} \tag{8}$$

where v_{mean} is the mean wind speed and κ is generally a function of altitude, which we take to be a constant. The velocity spectrum is given by the square root of the power spectrum. We therefore modulate a random Gaussian noise field $G(f)$ with the velocity spectrum to compute the spectrum of a particular (random) wind velocity field:

$$V(f) = G(f)\sqrt{P_V(f)} \tag{9}$$

The force due to the wind is complicated by the presence of turbulence [Feynman et al. 1964, Fig. 41-4], but can be generally modeled as a drag force proportional to the squared wind velocity. However, in our experiments, we have found that making the wind force directly proportional to wind velocity produces more pleasing results.

Finally, we assemble Equations (6)-(9) to construct the spectrum of the tip displacement $D_{\text{tip}}(f)$, take the inverse Fourier transform to generate the tip displacement $d_{\text{tip}}(t)$, and distribute the displacement over the branch according to equation (3). We apply the displacement as a rotation of each point about the root position of the branch. The displacements of points in the layer away from the motion armature are given by the displacement of the point on the armature that is the same distance from the root.

The user can control the resulting motion appearance by independently changing the mean wind speed v_{mean} and the natural (oscillatory) frequency f_o, mass m, and velocity damping term γ of each branch.

3.3 Water

Water surfaces belong to another class of natural phenomena that exhibit oscillatory responses to natural forces like wind. In this section we describe how one can specify a 3D water plane in a photograph and then define the mapping of water height out of that plane to displacements in image space. We then describe how to synthesize water height variations, again using a spectral method.

The motion armature for water is simply a plane; we assume that the image plane is the *xy plane* and the water surface is the *xz plane*. To correctly model the perspective effect, the user roughly specifies where the plane is. This perspective transformation M can be fully specified by the focal length and the tilt of the camera, which can be visualized by drawing the horizon [Criminisi et al. 2000].

After specifying the 3D water plane, the water is animated using a time-varying height field $h(q,t)$, where $q = (x_q, y_0, z_q)^T$ is a point on the water plane, and $y_0 = 0$ is the elevation of the water plane. To convert the height field h to the displacement map $d(p,t)$, for each pixel p we first find that pixel's corresponding point $q = Mp$ on the water plane. We then add the synthesized height $h(q,t)$ as a vertical displacement, which gives us a point $q' = (x_q, h(q,t), z_q)^T$. We then project q' back to the image plane to get $p' = M^{-1}q'$. The displacement vector for $d(p,t) = p' - p$ is therefore

$$d(p,t) = M^{-1}[Mp + (0, h(Mp,t), 0)^T] - p \tag{10}$$

Note that p and p' are affine points, d is a vector, and M is a 3×3 matrix.

The above model is technically correct if we want to displace objects on the surface of the water. In reality, the shimmer in the water is caused by local changes in surface normals. Therefore, a more physically realistic approach would be to use *normal* mapping, i.e., to convert the surface normals computed from the spatial gradients of $h(q,t)$ into two-dimensional displacements of the reflected rays. However, we have found that applying this normal mapping approach without a 3-dimensional model of the surrounding environment produces confusing distortions compared to our current approach, which generally produces pleasing, realistic-looking reflections as long as the wave amplitude is relatively small.

To synthesize a time-varying height field for the water, we use the user-specified wind velocity to synthesize a height field matching the statistics of real ocean waves, as described by Mastin *et al.* [1987]. Note that this approach deals only with ocean waves, which are gravity waves. Although it does not physically describe short-length waves, non-wind-generated waves on rivers/brooks/streams or large waves on shallow water, it gives plausible results for our application.

The spectrum filter we use for waves is the *Phillips spectrum* [Tessendorf 2001], which is a power spectrum describing the expected square amplitude of waves across all spatial frequencies $\mathbf{s}$

$$P_H(\mathbf{s}) \sim \frac{e^{\left[-1/(sL)^2\right]}}{s^4} \left|\hat{\mathbf{s}} \cdot \hat{\mathbf{v}}_{\text{mean}}\right|^2 \tag{11}$$

where $s = |\mathbf{s}|$, and $L = v_{\text{mean}}^2/g$, and g is the gravitational constant, and $\hat{\mathbf{s}}$ and $\hat{\mathbf{v}}_{\text{mean}}$ are normalized spatial frequency and wind direction vectors in the xz plane, respectively. (We denote 2D vectors in boldface.)

The square root of the power spectrum describes the amplitude of wave heights, which we can use to filter a random Gaussian noise field $G(\mathbf{s})$:

$$H_0(\mathbf{s}) = aG(\mathbf{s})\sqrt{P_H(\mathbf{s})} \tag{12}$$

where a is a constant of proportionality and H_0 is an instance of the height field which we can now animate by introducing time-varying phase. However, waves of different spatial frequencies move at different speeds. The relationship between the spatial frequency and the phase velocity is described by the well-known dispersion relation,

$$\omega(s) = \sqrt{gs} \tag{13}$$

The time varying height spectrum can thus be expressed as

$$H(\mathbf{s},t) = H_0(\mathbf{s})e^{i\omega(s)t} + H_0^*(-\mathbf{s})e^{-i\omega(s)t} \tag{14}$$

where H_0^* is the complex conjugate of H_0 [Tessendorf 2001]. We can now compute the height field at time $h(q,t)$ as the two-dimensional inverse Fourier transform of $H(\mathbf{s},t)$ with respect to spatial frequencies $\mathbf{s}$. We take the generated height field and tile the water surface using a scale parameter, β, to control the spatial frequency.

To recap the process, given the wind speed and direction, we synthesize a spectrum filter using equation (11) and apply it to a spatial Gaussian noise field to obtain an initial height field (12). This height field is then animated using equation (14) to synthesize the Fourier transform $H(\mathbf{s},t)$ of the height field $h(q,t)$ at time t. Taking the inverse Fourier transform, we recover the height field, use it to tile the water plane and substitute it into equation (10) to synthesize motion texture d_i at time t.

There are thus several motion parameters related to water: wind speed, wind direction, the size of the tile N, the amplitude scale a, and the spatial frequency scale β. The wind speed and direction are controlled globally for the whole animation. We find that a tile of size $N = 256$ usually produces nice looking results for the sizes of images we used. Users can change a to scale the height of the waves/ripples. Finally, scaling the frequencies by β changes the scale at which the wave simulation is being done. Simulating at a larger frequency scale gives a rougher look, while a smaller scale gives a smoother look. Hence, we call β the *roughness* in our user interface.

3.4 Boats

We approximate the motion of a bobbing boat by a 2D rigid transformation composed of a translation for heaving and a rotation for rolling. A boat moving on the surface of open water is almost always in oscillatory motion [Sun et al. 2003]. Hence, the simplest model is to assign a sinusoidal translation and a sinusoidal rotation. However, this often looks fake. In principle, we could build a simple model for the boat, convert the height field of water into a force interacting with the hull, and solve the dynamics equation for the boat to estimate its displacement. However, since our goal is to synthesize a quickly computable solution, we directly use the height field of the wave to move the boat, as follows.

We let the user select a line close to the bottom of the boat. Then, we sample several points q_i along the line and assume these points are on the water plane surrounding the boat. At time t, for each point q_i, we look up its displacement vector $d(p_i, t)$ (10) and calculate the corresponding position p_i' of p_i at time t as $p_i + d(p_i, t)$. Finally, we use linear regression to fit a line through the displaced positions. The position and orientation of the fitted line then determine the heaving and rolling of the boat.

3.5 Clouds

Another common element for scenic pictures is clouds. In principle, clouds could also be modeled as a stochastic process. However, we need the stochastic process to match the clouds in the image at some point, which is harder. Since clouds often move very slowly and their motion does not attract too much attention, we simply assign a translational motion field to them. We extend the clouds outside the image frame to create a cyclic texture using our inpainting algorithm, since their motion in one direction will create holes that we have to fill.

4 Results

We have developed an interactive system that supports matting, inpainting, motion editing, and previewing the results. We have applied our system to several photographs and famous paintings. The accompanying video provides a sense of the user interface for creating the animated pictures, as well as a demonstration of the animated results.

Table 1 summarizes the number of layers of each type created for the five animated pictures shown in Figure 1, the motion specification, along with the time that it took a user to perform the matting and in-painting steps (which are interleaved in the process, and thus difficult to separate in time), and the playback speeds. Generally the matting and in-painting steps take the large majority of the time. In all cases, the animated paintings take from a little under an hour to a few hours to create. Note that two of the animated pictures whose timings are presented above, "Boat Studio" and "Sunflowers," were created by a complete novice user who only had a few minutes of instruction before beginning work on the pictures. We provide playback speeds for our current unoptimized software implementation: Our code presently takes no special advantage of graphics hardware, but all of the operations could be readily mapped to GPUs, thereby greatly increasing frame rates.

For the Japanese Temple (Figure 1(a)), we model a total of 10 branches on the left and the right. We use a small wave amplitude ($a = 1.0$) and high roughness ($\beta = 200$) to give the ripples a fine-grained look. For the harbor picture in Figure 1(b), we animate the water and have nine boats swing with the water. The cloud and sky are animated using a translational motion field.

Figure 1(c)-(e) shows three paintings we have animated. Our technique works reasonably well with paintings, probably because in this situation we are even less sensitive to anything that does not look perfectly realistic. For Claude Monet's painting in Figure 1(c), we animate the water with lower amplitude roughness to keep the strokes intact. We also let the boat sway with the water. Another of Monet's paintings, shown in Figure 1(d), is a more complex example, with more than twenty layers. We use this example to demonstrate that we can change the appearance of the water by controlling the physical parameters. In Figure 3, we show the appearance of the water under different wind speeds, directions, and simulation scales.

For Van Gogh's sunflower painting (Figure 1(e)), we use our stochastic wind model to animate the twenty-five plant layers. With a simple sinusoidal model, the viewer usually can quickly figure out that the plants swing in synchrony, and the motion loses a lot of its interest. With the stochastic wind model, the flowers' motions decorrelate in phase and the resulted animation is more appealing. We also experimented with a very small amount of scaling along the branch armature in order to simulate foreshortening of the flowers as they move in and out of the image plane.

5 Conclusion and future work

In this paper, we have described an approach for animating still pictures of outdoor scenes that contain dynamic elements that respond to natural forces in a simple quasi-periodic fashion. We see our work as just a first step in the larger problem of animating a much more general class of pictures.

Before we began this work, it was not at all clear whether it would be possible to make still images come to life as animated scenes. We believe our judicious selection and enhancement of recently developed matting, inpainting, stochastic motion synthesis, image warping, and compositing algorithms provides an effective and easy-to-use system for generating realistic animations from static images.

We point out that our choice of techniques is especially well-suited to this problem, in that a relatively high-quality composite animation can be produced even when the results of each automated step are of objectively lower quality. First, the use of matting produces layers that are color-coherent along their boundaries, even if the resulting matte does not follow object boundaries. When in motion, these layers often seem perceptually plausible even when technically incorrect. Second, the limited amount of displacement we seek to introduce implies that the inpainting process can be relatively low-quality and still produce seamless composites. This allows us to use heuristic measures to reduce the search space and speed up the inpainting process. Finally, we do not ask end users to keyframe animations, but rather influence the scene in physical, easily understood terms, such as wind speed and direction. We provide a user interface that is accessible to users at all levels. Many users are already familiar with matting and inpainting processes from commercial products such as Photoshop, and the additional burden of assigning "canned" motion types is minimal.

Our system currently makes a number of assumptions that we would like to relax. For example, we assume that the elements of the input image are in their equilibrium positions. This is often not the case, e.g., for a scene with water that already has ripples. Indeed, an interesting challenge would be to use these ripples to estimate the water motion, unwarp the reference image and then animate it correctly. In addition, we currently ignore the effects of shadows, transparency, and reflections. For example, the reflections of the boat move with the deformations of the water, but do not account for any additional motion due to the boat's bobbing up and down. When the motion is large, the results are less realistic. One solution would be to segment out reflections, transparent layers and shadows somehow, and let them move with the casting objects accordingly.

Many of our approximations limit the plausibility of very large-scale motions, in which pixels are warped more than a few dozen pixels from their source position. For example, we simulate boats rolling as a 2D rigid motion. It might be possible to fake a slight 3D rotation with a non-rigid distortion, to allow for more plausible

(a) composite (b) lower wind speed (c) wind of different direction (d) rougher water surface

Figure 3 We can control the appearance of water surface by adjusting some physical parameters such as wind speed. We show one of the composites (a) as the reference, in which the wind blow at 5 m/s in z direction. We decrease the wind speed to 3 m/s (b) and change the wind direction to be along z axis (c). In (d), we change the scale of the simulation to render water with finer ripples.

	Trees	*Water*	*Boats*	*Clouds*	*Still*	*Layering*	*Animating*	*Rendering*	*Resolution*
Japansese Temple	10	1	0	0	2	45 m	10 m	7 fps	900x675
Harbor	0	2	9	1	5	90 m	10 m	3.8 fps	900x600
Boat Studio	0	1	1	0	1	30 m	10 m	10 fps	600x692
Argenteuil	16	1	3	1	3	120 m	15 m	4.1 fps	800x598
Sunflowers	25	0	0	0	1	210 m	20 m	5.1 fps	576x480

Table 1 The number of layers of each type for each of the five examples in Figure 1, along with approximate times in minutes for a user to perform the layering steps (including matting and inpainting), animating step (including motion specification and editing), and playback speeds.

large-scale motions. Very large warps of the water surface can appear distorted due to warping from outside the image boundaries, and when the water waves become large enough under very windy conditions, we expect to see a number of additional real-world effects such as water "lapping up" against the shore or boats, "whitecaps," splashes, or other turbulent surface effects.

Our method currently works best for trees at a distance. For nearby trees, it is presently difficult and tedious to segment the leaf and branch structure properly. It would also be interesting to add the "shimmering" effect of leaves blowing in the wind by applying turbulent flow fields within the tree layers.

There are other classes of motion that could be modeled using a similar approach. We imagine that waterfalls, ocean waves, flying birds and other small animals, flame, and smoke may all be possible. For example, waterfalls could perhaps be animated using a technique similar to "motion without movement" [Freeman et al. 1991]. Ocean waves could be simulated using stochastic models, although matching the appearance of the source image poses some interesting challenges. Flying birds and other small animals could be animated using ideas from video sprites [Schödl et al. 2000]. We believe that it might also be possible to animate fluids like flame or smoke. However, this would require a constrained stochastic simulation, since the state of simulation should resemble the appearance of the input image. Recent advances in controlling smoke simulation by keyframes could be used for this purpose [Treuille et al. 2003].

In our system, all the layers are hooked up together to a synthetic wind force. Currently, the same mean wind velocity is applied everywhere in the scene. It would be straightforward to extend the formulation to handle complete vector fields of evolving wind forces in order to provide a more realistic style of animation such as moving gusts of wind. In addition, we could add more controllability so that the users could interact with trees individually.

Currently, we use physically-based simulation to synthesize a parametric motion field, but the quality of the motion could potentially be improved using learning algorithms to transfer motion from similar type of objects in videos.

Furthermore, our motion model addresses only a restricted range of motions. We imagine future systems might handle transitions between different types of motion, animation to or from a rest state, water features such as streams that move continuously in a single direction, and transitions between different scene states and/or types of motion (e.g. weather changing from calm to stormy, skies changing from clear to cloudy, boats traveling to and from the horizon, etc.).

Our system presently requires a fair amount of user interaction. We hope to further reduce the time and effort to create these animations by exploiting continued advances in intelligent image selection and matting algorithms such as GrabCut [Rother et al. 2004] or Lazy Snapping [Li et al. 2004]. Furthermore, an automated or semi-automated region classification to identify features such as foreground tree branches and water would enable a much more automated process. For example, one could imagine automatically identifying the "white water" of a waterfall, and then automatically animating the waterfall. For a lake with a simple boundary, such as in Figure 1(a), it might also be possible to automatically segment the the water region by identifying reflections.

Another possibility would be to use multiple pictures as input. Most modern digital cameras have a "motor-drive" mode that allows users to take high-resolution photographs at a restricted sampling rate, around 1–3 frames per second. From such a set of photographs we might be able to automatically segment a picture into several coherently moving regions and figure out the motion parameters from the sample still images. It would also be interesting to combine high-resolution stills with lower-resolution video to produce attractive animations. Our approach could also be combined with "Tour into the picture" to provide an even richer experience, with the ability to move the camera and less constrained perspective planes.

In conclusion, we have shown the ease with which it is possible to breathe life into pictures, based on recently developed matting, inpainting, and stochastic modeling algorithms. We hope that our work will inspire other to explore the creative possibilities in this rich domain.

Acknowledgments

The authors wish to thank Wil Li for narrating our video, and Mira Dontcheva for user-testing our segmentation and inpainting sytem. We would also like to thank the reviewers for their helpful comments. This work was supported by the University of Washington Animation Research Labs, Washington Research Foundation, NSF

grant CCR-0098005, NSC 94-2213-E-002-051, NSC 93-2622-E-002-033 and an industrial gift from Microsoft Research.

References

AOKI, M., SHINYA, M., TSUTSUGUCHI, K., AND KOTANI, N. 1999. Dynamic texture: Physically-based 2D animation. In *ACM SIGGRAPH 1999 Conference Sketches and Applications*, 239.

BARRETT, W. A., AND CHENEY, A. S. 2002. Object-based image editing. *ACM Transactions on Graphics 21*, 3, 777–784.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Proceedings of ACM SIGGRAPH 2000*, 417–424.

CHUANG, Y.-Y., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2001. A Bayesian approach to digital matting. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2001*, vol. II, 264–271.

CRIMINISI, A., REID, I. D., AND ZISSERMAN, A. 2000. Single view metrology. *International Journal of Computer Vision 40*, 2, 123–148.

CRIMINISI, A., PEREZ, P., AND TOYAMA, K. 2003. Object removal by exemplar-based inpainting. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2003*, vol. II, 721–728.

DRORI, I., COHEN-OR, D., AND YESHURUN, H. 2003. Fragment-based image completion. *ACM Transactions on Graphics 22*, 3, 303–312.

FEYNMAN, R. P., LEIGHTON, R. B., AND SANDS, M. 1964. *The Feynman Lectures On Physics, Volume II: Mainly Electromagnetism and Matter*. Addison Wesley, Reading, Mass.

FREEMAN, W. T., ADELSON, E. H., AND HEEGER, D. J. 1991. Motion without movement. *Computer Graphics (Proceedings of ACM SIGGRAPH 91) 25*, 4, 27–30.

GRIFFITHS, D., 1997. Lake java applet. http://www.jaydax.co.uk/tutorials/laketutorial/dgclassfiles.html.

HATHAWAY, T., BOWERS, D., PEASE, D., AND WENDEL, S., 2003. http://www.mechanicalmusicpress.com/history/pianella/p40.htm.

HORRY, Y., ANJYO, K.-I., AND ARAI, K. 1997. Tour into the picture: using a spidery mesh interface to make a nimation from a single image. In *Proceedings of ACM SIGGRAPH 1997*, 225–232.

JIA, J., AND TANG, C.-K. 2003. Image repairing: Robust image synthesis by adaptive ND tensor voting. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2003*, vol. I, 643–650.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics 21*, 3, 465–472.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *ACM Transactions on Graphics 23*, 3, 303–308.

LITWINOWICZ, P., AND WILLIAMS, L. 1994. Animating images with drawings. In *Proceedings of ACM SIGGRAPH 1994*, 409–412.

MASTIN, G. A., WATTERBERG, P. A., AND MAREDA, J. F. 1987. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications 7*, 3, 16–23.

MORTENSEN, E. N., AND BARRETT, W. A. 1995. Intelligent scissors for image composition. In *Proceedings of ACM SIGGRAPH 1995*, 191–198.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proceedings of ACM SIGGRAPH 2001*, 433–442.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. *Computer Graphics (Proceedings of ACM SIGGRAPH 84) 18*, 4, 253–259.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. Grabcut — interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics 23*, 3, 309–314.

RUZON, M. A., AND TOMASI, C. 2000. Alpha estimation in natural images. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2000*, 18–25.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of ACM SIGGRAPH 2000*, 489–498.

SHADE, J., GORTLER, S., HE, L.-W., AND SZELISKI, R. 1998. Layered depth images. In *Proceedings of ACM SIGGRAPH 1998*, 231–242.

SHINYA, M., AND FOURNIER, A. 1992. Stochastic motion – motion under the influence of wind. *Computer Graphics Forum 11*, 3, 119–128.

SHINYA, M., MORI, T., AND OSUMI, N. 1998. Periodic motion synthesis and Fourier compression. *The Journal of Visualization and Computer Animation 9*, 3, 95–107.

SIMIU, E., AND SCANLAN, R. H. 1986. *Wind Effects on Structures*. John Wiley & Sons.

SOATTO, S., DORETTO, G., AND WU, Y. N. 2001. Dynamic textures. In *Proceedings of IEEE International Conference on Computer Vision (ICCV) 2001*, 439–446.

STAM, J., AND FIUME, E. 1993. Turbulent wind fields for gaseous phenomena. In *Proceedings of ACM SIGGRAPH 1993*, 369–376.

STAM, J. 1995. *Multi-Scale Stochastic Modelling of Complex Natural Phenomena*. PhD thesis, Dept. of Computer Science, University of Toronto.

STAM, J. 1997. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum 16*, 3, 159–164.

SUN, M., JEPSON, A. D., AND FIUME, E. 2003. Video input driven animation (VIDA). In *Proceedings of IEEE International Conference on Computer Vision (ICCV) 2003*, 96–103.

SZUMMER, M., AND PICARD, R. W. 1996. Temporal texture modeling. In *Proceedings of IEEE International Conference on Image Processing (ICIP) 1996*, vol. 3, 823–826.

TESSENDORF, J. 2001. Simulating ocean water. *ACM SIGGRAPH 2001 course notes No. 47 Simulating Nature: Realistic and Interactive Techniques*.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Trans. Graph. 22*, 3, 716–723.

WANG, Y., AND ZHU, S. C. 2003. Modeling textured motion: Particle, wave and sketch. In *Proceedings of IEEE International Conference on Computer Vision (ICCV) 2003*, 213–220.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, 479–488.