Eurographics Symposium on Rendering (2007)
Jan Kautz and Sumanta Pattanaik (Editors)

Feature-Guided Dynamic Texture Synthesis on Continuous Flows

Rahul Narain[1], Vivek Kwatra[1], Huai-Ping Lee[1], Theodore Kim[2], Mark Carlson[3], and Ming C. Lin[1]

[1]University of North Carolina at Chapel Hill
[2]IBM TJ Watson Research Center
[3]Walt Disney Animation Studios

Abstract

We present a technique for synthesizing spatially and temporally varying textures on continuous flows using image or video input, guided by the physical characteristics of the fluid stream itself. This approach enables the generation of realistic textures on the fluid that correspond to the local flow behavior, creating the appearance of complex surface effects, such as foam and small bubbles. Our technique requires only a simple specification of texture behavior, and automatically generates and tracks the features and texture over time in a temporally coherent manner. Based on this framework, we also introduce a technique to perform feature-guided video synthesis. We demonstrate our algorithm on several simulated and recorded natural phenomena, including splashing water and lava flows. We also show how our methodology can be extended beyond realistic appearance synthesis to more general scenarios, such as temperature-guided synthesis of complex surface phenomena in a liquid during boiling.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction techniques I.4.8 [Image Processing and Computer Vision]: Scene Analysis – Motion

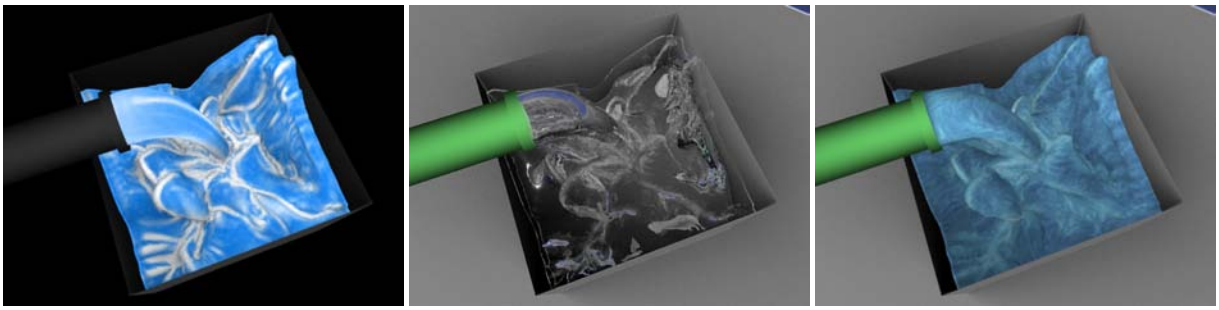
1. Introduction

The realistic simulation and rendering of continuous flow patterns in nature, such as liquids, viscous flows, flames and smoke, has been an important but challenging problem in computer graphics. Real flows exhibit complex surface behaviors, such as small ripples in a stream, foam and bubbles in turbulent flow, and crust patterns in lava, which cannot be easily reproduced by traditional simulation techniques. The visual appearance of these phenomena can be rendered as dynamic textures over the flow surfaces. These textures move with the flow on the dynamically changing fluid surface and vary over space and time in correlation with the underlying physical characteristics of the flow. Existing techniques do not generate such evolving, heterogeneous textures over arbitrary flow surfaces in a physically consistent and visually convincing manner.

Main Results: We propose a novel technique in which the physical and geometric characteristics of the flow are used to intelligently guide the synthesis of dynamic, non-uniform textures over its domain. These characteristics are derived from the surface properties and velocity fields of continuous flows, obtained from either a simulated liquid animation or pre-recorded video footage. These characteristics, which we term as *features*, are used to model the behavior of surface flow phenomena over space and time. This model then automatically controls the synthesis of a spatially and temporally varying texture over each frame of the simulation or video clip using multiple, dissimilar input textures, as shown in Fig. 1.

To our knowledge, this is the first technique which generates complex physically-based flow appearance through texture synthesis. We combine physical and geometric features, controllable texture synthesis on continuous flows, and video textures, using a single optimization-based texture synthesis framework. This framework for feature-guided texture synthesis is a very general approach, capable of producing visually convincing results for diverse physical and visual effects with little user guidance.

The rest of the paper is organized as follows: In Section 2, we summarize the related work in relevant areas. Section 3



(a) (b) (c)

Figure 1: *(see color plate) Water pours from a hose into a box. Our technique generates a dynamic, spatially varying texture that reproduces small-scale phenomena such as foam and ripples on the liquid surface. Using physical characteristics of the flow, we compute the feature map shown in (a) to guide texture synthesis, which allows us to generate different visual effects as shown in (b) and (c). These and other results in this paper are animated and can be seen in motion in the supplementary video.*

is the heart of this paper, where we present all the various aspects of feature-guided synthesis. In Section 4, we describe our method for using real videos to capture texture evolution, in order to perform feature-guided video synthesis. Finally, we present our results for a variety of different scenarios and effects in Section 5, and conclude in Section 6.

2. Related work

Texture synthesis has been extensively researched in the context of novel image and video creation and manipulation. A comprehensive review here is impossible, but we provide a brief summary of the most relevant prior literature. A pertinent extension of traditional texture synthesis methods is the use of flow fields to guide the motion of texture elements for synthesizing animated textures. This capability was demonstrated by Stam [Sta99] who advected solid textures through a fluid flow field, and Neyret [Ney03], who pioneered the self-regenerating advection of unstructured textures over flow fields in 2D and 3D. Newer approaches are able to perform this on more complex textures in the 2D domain [BSHK04, KEBK05]. Previous work on lava simulation [SAC*99] generated procedural lava textures by tracking particles on the fluid surface across frames. Recently, a few authors have proposed techniques for advection of more general textures on animated 3D surfaces [KAK*07, BSM*06, HZW*06]. These works focus on generating homogeneous textures that do not necessarily exhibit meaningful variation over space and time.

The generation of heterogeneous, spatially varying textures has been demonstrated in 2D and on 3D surfaces. In fact, many texture synthesis methods in 2D [EF01, Ash01, KSE*03, KEBK05] and 3D [ZZV*03] are capable of performing controllable texture synthesis according to user-specified constraints. The use of an explicitly specified feature map for controlling texture synthesis was introduced in the context of texture transfer [HJO*01, EF01]. However, they do not address how to determine the feature map itself. The idea of generating a feature map automatically using geometric characteristics of static meshes has been proposed very recently [MKC*06]. Salient features can also be computed on the input textures, which can improve synthesis quality [WY04] and allow blending between textures [ZZV*03].

Video synthesis has also been widely investigated. Some methods focus on manipulation of video, treating a full frame as a unit [SSSE00, DS03]. [WL00, KSE*03] treat video texture synthesis as a 3D texturing problem, producing a spatio-temporal volume. Another technique that bears some similarity to what we propose is the work of [BSHK04], where they track the evolution of texture patches along stationary flow lines in the input and render them with the desired motion into the output video. However, these approaches do not easily extend to synthesis on dynamically changing fluid surfaces.

The distinguishing characteristic of our work, with respect to the related work described here, is that we *automatically* extract features from highly dynamic fluid phenomena, which besides providing the relevant properties for texture synthesis in every frame, also vary in a *temporally smooth* and consistent manner over time. Additionally, we provide a *unified optimization-based framework* for feature-guided texture synthesis that allows any new feature to be incorporated without much extra effort, and can handle both video as well as image textures.

3. Feature-Guided Synthesis

This work builds on existing optimization-based frameworks for texturing of surfaces and fluids [KAK*07, BSM*06, HZW*06], which synthesize high-quality textures on fluid surfaces from input texture exemplars. In contrast to these

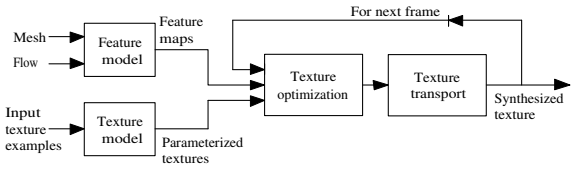


Figure 2: *An overview of our technique for feature-guided texture synthesis on flows.*

approaches, we are able to automatically extract visually salient features using physical characteristics and geometric properties to model temporally and spatially varying patterns of complex natural phenomena.

The basic concept of our approach is based on the observation that texture elements on a flow surface, such as ripples and foam on turbulent water, show variation on both large and small scales of space and time. Traditional texture synthesis only takes into account the small-scale spatial variation of the texture. The large-scale variation of texture on real flows is typically correlated with the physical characteristics of the flow. On fluids, texture also displays small-scale evolution over time which cannot be captured by using static images as texture exemplars. We propose a general framework which integrates flow features and video exemplars seamlessly in the optimization-based texture synthesis technique.

The main elements of our system are shown in Fig. 2. We take as input a sequence of animated fluid surface meshes and the time-varying velocity field of the flow. For each frame, we use the physical features of the fluid to compute a *parameter map* over the surface, which defines the spatial variation of texture appearance. This parameter map is used to guide the texture synthesis process on the surface based on the texture and video input. The texture and the parameter map are then transported via the flow to the corresponding locations on the next frame to be used for maintaining temporal coherence.

3.1. Synthesis on Fluids

We briefly review the existing approach for synthesis on fluids using homogeneous textures as input exemplars. For a flow sequence, texture synthesis is performed one frame at a time, treating the flow surface at a particular frame as a static mesh. As a preliminary step, the surface is covered uniformly with overlapping patches to facilitate comparison with the 2D input texture. The output texture is then generated by minimizing an energy function $E(\mathbf{s};\mathbf{p})$ which measures the similarity of the texture at each surface patch $\mathbf{s}$ with a similar patch $\mathbf{p}$ in the input exemplar.

For temporal coherence, the generated texture from the previous frame is advected using the known velocity field over time to yield a target texture, which is used as a soft constraint in the texture optimization step. Thus the net energy function for a surface patch $\mathbf{s}$ is

$$E(\mathbf{s}) = E_{texture}(\mathbf{s};\mathbf{p}) + E_{coherence}(\mathbf{s};\mathbf{s}')$$

where $\mathbf{s}'$ is the advected texture color. These energy functions measure the intensity/color discrepancy between the surface patch and the corresponding input and advected patches respectively, and generally take the following form:

$$\begin{aligned} E_{texture}(\mathbf{s};\mathbf{p}) &= ||\mathbf{I}(\mathbf{s}) - \mathbf{I}(\mathbf{p})||^2 \\ E_{coherence}(\mathbf{s};\mathbf{s}') &= ||\mathbf{I}(\mathbf{s}) - \mathbf{I}(\mathbf{s}')||^2 \end{aligned}$$

where $\mathbf{I}$ refers to the intensities or colors associated with a patch. The total energy over all patches, $\sum_{\mathbf{s}} E(\mathbf{s})$, can be minimized by an iterative approach; we refer the reader to previous work for details [KEBK05, KAK*07, BSM*06]. Additionally, techniques such as multi-resolution synthesis [WL00] and appearance-space transformation [LH06] can be used to improve synthesis quality within this framework.

The process of constructing local patches on the surface for texture synthesis requires a smooth local orientation field at each point on the surface. This field also controls the orientation of the output texture. To maintain temporal coherence of texture orientation, the orientation field is advected across frames using a modified advection procedure [KAK*07].

The texture synthesis technique described above is only capable of generating textures which are roughly homogeneous over space and time. While two or more textures can be used (as in [BSM*06]) this approach is limited to assignment at the initial state, beyond which they cannot be varied dynamically. The result is that while the synthesized textures show temporal coherence and correctly move with the fluid flow, they are too uniform over space and time to convincingly display dynamically evolving textures. Next we present our contribution for synthesis of spatially varying textures.

3.2. Flow Features

We control the spatial and temporal variation of texture using the physical features of the flow itself. We use some metrics to characterize such properties of the flow, which we term *features*. The values of these features over the flow surface can then be used to guide the texture synthesis process. The choice of metric typically depends on the type of flows and the specific effect to be generated.

Basic Principles: We have considered and evaluated several possible operators for controlling the texture synthesis. One can use arbitrary combinations of such metrics to allow the various kinds of effects to be simulated. Experimentation reveals, however, that better results are obtained from features which are physically motivated by the physics of the desired effects rather than from *ad hoc* metrics. Below, we

discuss specific examples of such features which we found generally useful for applications of fluid texture synthesis.

Curvature: The mean surface curvature of the fluid surface is commonly used as a measure for adding splashes and foam to water simulations [FR86, KCC*06]. Therefore, we use this as the natural metric to simulate foam through our texture synthesis approach. This feature is described further in Section 3.4.

Divergence: For simulating viscous fluids such as lava flow, the visually significant behavior is the anisotropic stretch and squash of the fluid surface. This behavior causes effects such as cracking and crumpling of the lava crust. One metric which can be used in this case is the divergence of the velocity flow. Since we are only interested in the surface behavior of the flow, we project the divergence operator onto the tangent plane at the surface, so that

$$\nabla_t \cdot \mathbf{u} = D_{\mathbf{e}_1}(\mathbf{e}_1^T \mathbf{u}) + D_{\mathbf{e}_2}(\mathbf{e}_2^T \mathbf{u})$$

where D_{e_i} denotes the directional derivative along orthogonal tangent vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ at the surface. This is also useful since for an incompressible flow, $\nabla \cdot \mathbf{u}$ is identically zero while $\nabla_t \cdot \mathbf{u}$ may not be.

Jacobian: The Jacobian matrix of the velocity field is another useful characteristic which is used in, for example, flow visualization [PVH*03]. Again, we consider only the tangential component $\mathbf{J}_t$:

$$\mathbf{J}_t(\mathbf{u}) = [D_{\mathbf{e}_j}(\mathbf{e}_i^T \mathbf{u})]_{2\times 2} = [\mathbf{e}_1 \mathbf{e}_2]^T \mathbf{J} [\mathbf{e}_1 \mathbf{e}_2]$$

$\mathbf{J}_t$ completely characterizes the local spatial variation of the velocity in the tangent plane. In particular, the eigenvalues of its symmetric component, defined as

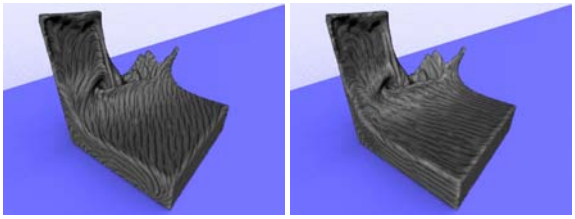
$$\mathbf{J}_t^+ = \frac{1}{2}(\mathbf{J}_t + \mathbf{J}_t^T),$$

measure the extremal values of directional divergence. We found these eigenvalues to be a more useful measure of flow behavior than the divergence alone, especially for viscous flow, such as lava, where the directional stretching and compression of the fluid is relevant.

These are not the only useful features for controllable texture synthesis. Depending on the specific application, any other metric which can be evaluated on the fluid surface can be used. For example, for simulating lava, the surface temperature is an important feature which influences the appearance of the fluid. The user is free to include this as an additional feature, whether approximated as a function of position or evaluated through more accurate methods.

3.3. Orientation Control

It is desirable in the case of strongly directional flow to determine the orientation of the advected texture accordingly. This control becomes important when synthesizing anisotropic textures, since the direction of anisotropy is typically related to the characteristics of the flow. Examples include the patterns formed on the crust of flowing lava, which are caused by non-uniform stretching or compression of the surface in different directions.



(a) (b)

Figure 3: *Flow-based control of orientation. (a) Without orientation control, the orientations are arbitrary and do not reflect the flow characteristics. (b) Using flow features, we obtain a coherent orientation field which is correlated with the flow.*

In order to orient the texture patches along the appropriate directions, we require a field of tangential directions over the fluid surface. We propose that for texturing flows, the most important directions are extrema of directional divergence. Under the influence of the flow, a small linear element will tend to align over time with the direction of maximum divergence. The extrema are directly given by the eigenvectors of the tangential Jacobian matrix $\mathbf{J}_t$, since

$$\nabla_{\mathbf{e}} \cdot \mathbf{u} = \mathbf{e}^T D_{\mathbf{e}} \mathbf{u} = \mathbf{e}^T \mathbf{J}(\mathbf{u}) \mathbf{e} \in [\min eig(\mathbf{J}(\mathbf{u})), \max eig(\mathbf{J}(\mathbf{u}))]$$

In practice, we use the symmetric component $\mathbf{J}_t^+$ rather than the full tangential matrix $\mathbf{J}_t$, which ensures that the eigenvectors are real everywhere and orthogonal to each other. Since these eigenvectors are unstable at regions where the eigenvalues are small and/or nearly equal, we do not directly set the orientation field $\mathbf{d}$ at each point to be equal to the maximal eigenvector, say $\mathbf{v}$; instead, we apply a force to adjust its direction towards $\mathbf{v}(\mathbf{x})$ over time:

$$\frac{\partial \mathbf{d}}{\partial t} = \mathbf{D}(\mathbf{d}, \mathbf{u}) + \lambda(\mathbf{I} - \mathbf{d}\mathbf{d}^T)\mathbf{v}$$

where $\mathbf{D}$ is the modified advection operator for transport of orientation fields [KAK*07]. The strength λ of the force is set to be proportional to the anisotropy of the flow, which we measure through the stability of $\mathbf{v}$ relative to perturbations of $\mathbf{J}_t$. This adjustment ensures that the orientation field is governed by the maximal eigenvector only in regions where the flow is strongly anisotropic.

3.4. Feature-Guided Textures

Once we have determined the flow features used to vary the synthesized texture over the surface, we have to integrate them into the texture optimization process in order to be

able to generate non-uniform textures based on these features. Thus, it is necessary to relate the fluid features, which are continuous in nature, to a finite set of input textures.

We adopt the formulation used in existing work on texture transfer [HJO*01, EL99, MKC*06]. The input textures are augmented with a *parameter map* $\mathbf{r}$ (variously termed "feature map" and "correspondence map" in previous work) which describes the spatial variation of texture over the input. For example, a spatially varying foam texture can be parameterized by the density of foam, while a texture representing lava could vary according to surface temperature and the solidity of the surface crust. Then, texture synthesis can be controlled by defining the target parameter values over the output domain. This target parameter map guides the synthesis process to favor the selection of texture patches, which have similar values as the ones on each surface pixel. It is straight-forward to incorporate such an approach in the texture optimization framework: we simply introduce an additional energy function

$$E_{feature}(\mathbf{s};\mathbf{p}) = ||\mathbf{r}(\mathbf{s}) - \mathbf{r}(\mathbf{p})||^2$$

which measures the similarity of feature values on the surface to those of the selected input patches.

Our contribution in this part lies in relating this parameter map to the feature values obtained from the physical behavior of the fluid. Note that there is a semantic gap between the feature values, say $\mathbf{q}$ in vectorized form, which are typically physical quantities, and texture parameters $\mathbf{r}$, which are more qualitative in nature. The relation between them depends on the application. For example, for turbulent water, relevant feature values include surface curvature, while texture parameters include the density of foam on the surface, with very high curvature values (such as splashes and breaking waves) leading to the increase of foam density.

While it is possible to set $\mathbf{r}$ directly as a function of the instantaneous feature values $\mathbf{q}$, we found that this does not yield convincing results as it lacks the proper temporal behavior: for example, foam would disappear immediately when the curvature decreased. Therefore, we allow for modeling the relationship between $\mathbf{q}$ and $\mathbf{r}$ as a general first-order differential equation relating the *change* in parameters to a function of the feature values:

$$\frac{D\mathbf{r}}{Dt} = f(\mathbf{q},\mathbf{r})$$

Here we use the Lagrangian derivative so that the parameter map is also transported with the flow in a realistic manner. In practice, we implement this by first advecting the parameter map via the flow field, and then updating its value using $f(\mathbf{q},\mathbf{r})$.

As we described in the previous sections, the texture properties relevant to the flow features may need to be determined on a case by case basis. While this aspect may be considered as a limitation if the goal is to achieve complete automation, we believe that it actually empowers the animator to make artistic choices while using our system and therefore enhances its usability. Note that the texture-feature relationship needs to be determined once for a given texture, and can be quite simple for static textures, such as the assignment of a discrete label such as "turbulent", "calm", "intermediate" and so on.

4. Video Textures

Many fluid phenomena exhibit meso-scale evolution over time, which is distinct from advection behavior. For example, foam and ripples on a water surface are not simply fixed to only move with the water's large-scale flow, but also change over time in a characteristic manner. To faithfully reproduce such behavior, we propose the use of videos of real fluids as exemplars for the texture synthesis process. Of course, in real videos the fluid will have its own flow which also transports the texture over time. We factor out this flow to obtain stationary input exemplars, as described below, and use a novel video texture model to generate new animated textures which show similar temporal behavior.

4.1. Motion Removal

To decouple meso-scale texture evolution from the fluid flow in the video, we determine the flow field of the input video using a motion estimation technique. There are two main methods for motion estimation, namely optical flow [SH81, BA96] and feature tracking [TK91, ST94, Low04]. We tested both approaches on sample videos of fluids, and found that optical flow worked consistently better for these examples. This finding may be due to the nature of fluid videos, which are largely homogeneous and have non-rigid motions.

The output from the motion estimation procedure yields a dense flow field which accurately captures the temporal changes in the video. However, since the motion estimation cannot distinguish between large-scale advection and texture evolution, both of these are reflected in the output motion field. Therefore, to remove the effect of the latter, we perform a Gaussian smoothing step on the motion field to retain only the large-scale motion of the fluid in the video as our flow estimate.

4.2. Synthesis of Evolving Textures

The technique we present below allows for video textures to be supported in the texture optimization framework, thus enabling such animated textures to be synthesized with high texture quality.

In order to synthesize an evolving texture whose behavior matches that of the input video, we require a model of how the input video changes over time, so that we can predict, given the current state of the texture, its likely appearance in the next frame. We build such a model implicitly by using

tuples of $n+1$ corresponding patches $[\mathbf{p}_n, \ldots, \mathbf{p}_1, \mathbf{p}_0]$ from consecutive frames of the video as samples of its temporal behavior. We refer to such tuples as *video patches*, since they are the basic unit of representation for video textures, analogous to 2D texture patches for static textures. We construct the set of video patches by selecting corresponding image patches on every set of $n+1$ consecutive frames, after tracking and warping to compensate for the estimated flow field. The number of previous frames n used determines the degree of continuity in the synthesized texture, and a value of 1 or 2 should be sufficient for most applications. A similar model was used in previous work [SSSE00] to correctly account for dynamical continuity.

For handling texture evolution on a surface, we represent the texture state by storing, at each vertex, not just the current color but tuples of the current and past color values over n frames. For synthesizing a new frame, recall that due to the transport procedure we know the texture colors from the previous frames, say $\mathbf{s}_{n+1}, \ldots, \mathbf{s}_1$. Since we want the evolution of a surface patch to closely match that of the input video patches, we define the energy function for the current texture of the surface patch $\mathbf{s}_0$ with respect to a selected video patch $\mathbf{p} = [\mathbf{p}_n, \ldots, \mathbf{p}_0]$ simply as

$$E(\mathbf{s};\mathbf{p}) = \sum_{i=0}^{n} w_i^2 (||\mathbf{I}(\mathbf{s}_i) - \mathbf{I}(\mathbf{p}_i)||^2 + ||\mathbf{r}(\mathbf{s}_i) - \mathbf{r}(\mathbf{p}_i)||^2)$$

where $w_n, \ldots, w_0$ are weights that can be used to control the relative importance of previous frames in the energy function. Texture synthesis over the entire surface is performed by minimizing the sum of this energy function over all patches, varying only the current texture $\mathbf{s}_0$.

Since this new energy function is also quadratic, it can still be solved using the same fast least-squares solvers used in existing work. Specifically, the texture synthesis approach still has the same two steps as before. In each iteration, we first find the input patch $\mathbf{p}$ for each surface patch $\mathbf{s}$ which most closely matches the previous (and current, if known) values, and then determine $\mathbf{s}_0$ for all surface patches to minimize the total energy $\sum_{\mathbf{s}} E(\mathbf{s};\mathbf{p})$.

This approach combines both the main texture similarity energy and the texture evolution constraint into a single logically consistent formulation, and automatically accounts for temporal coherence as well. For a given surface patch, the selected input video patch $\mathbf{p}$ chosen will be one which matches the previous patches $\mathbf{s}_n, \ldots, \mathbf{s}_1$. If the input patches are temporally coherent so that $\mathbf{p}_0$ is similar to $\mathbf{p}_1$, then $\mathbf{s}_0$ necessarily must be correspondingly close to the previous frame's $\mathbf{s}_1$. On the other hand, if the input video contains discontinuities between frames, then the texture will also behave similarly. We believe this is a desirable behavior, and that the fact that temporal coherence as a whole arises naturally from this formulation without an additional term is of significant interest.

While we have developed this procedure for the synthesis of video textures on fluid surfaces, it is not restricted to this particular application. It can be used for the synthesis of any time-varying texture, whether in the 2D domain or on 3D surfaces. Also, our new energy function can replace the original energy function even when only static image-based textures are used.

5. Results

We have implemented our technique in C++ and rendered the results using 3Delight®. We have applied it to several diverse scenarios with different fluids and input textures, demonstrating the variety of effects that can be produced by this technique. The supplementary video includes these results.

5.1. Image Textures

In Figure 4, a large quantity of water is dropped on a broken dam, causing the water to splash. Three input textures are used with varying amounts of foam. The associated texture parameter on the surface, representing foam density, is controlled by the surface curvature and decays gradually over time. An additional continuity term using $\nabla_t \mathbf{u}$ ensures that the total amount of foam is conserved under advection. Precisely,

$$\frac{Dr}{Dt} = f(H, \nabla_t \mathbf{u}; r) = k \cdot \max(\rho H - r, 0) - r/\tau - r\nabla_t \mathbf{u}$$

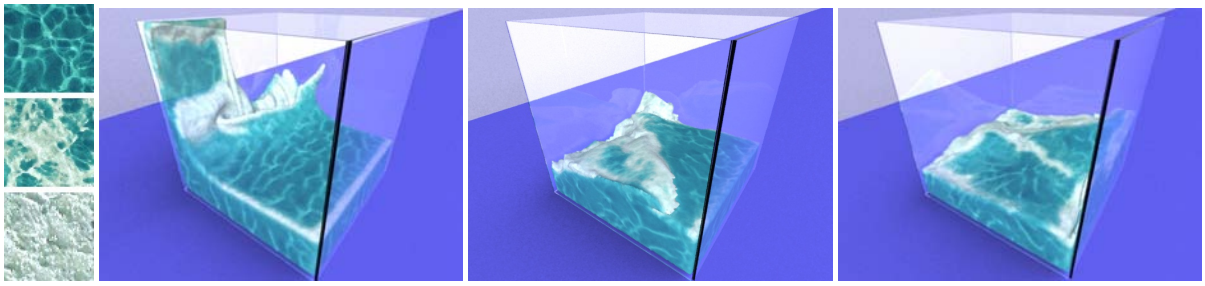
We clamped r to the range $[0,1]$, and used parameter values $\rho = 0.02$ m, $\tau = 2$ seconds, and $k \equiv \infty$ by directly assigning $r = \rho H$ if $r < \rho H$.

For comparison, in Figure 5 we show the results of texture synthesis when the fluid features are not taken into account. If features are not used, the texture is not correlated with the behavior of the fluid, and interestingly it converges to a local optimum of a uniform texture over time.

Figure 6 shows the different kinds of lava flow that can be generated by our technique. In the first row, we simulate hot molten lava cooling off over time, using a real image of lava with a user-generated feature map assigned. We use a spatially varying feature field which includes distance from the source to approximate temperature and velocity divergence to model the clumping visible in the input image. The second row shows a colder lava flow with a solidified crust, using manually designed inputs textures. Here the maximum value of directional divergence is used to model the cracking of the crust.

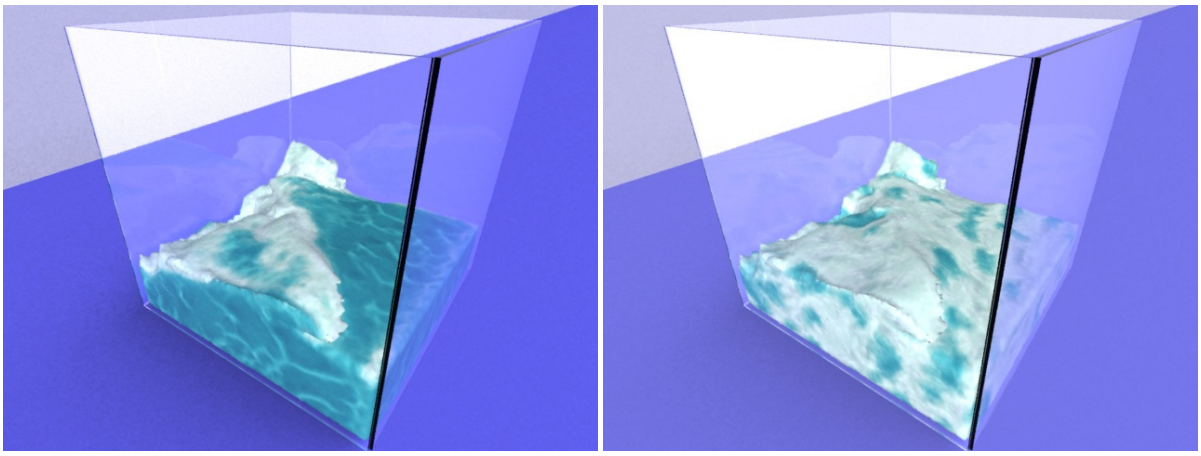
5.2. Video Textures

We have used video textures for Figure 7. We used only $n = 1$ in this examples; that is, only the immediately previous color is used in selecting the current color. The target features were determined in the same way as in Figure 4.



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

Figure 4: *(see color plate) (a) Texture inputs for water foam. (b, c, d) Three frames of the synthesized foam texture on the broken dam simulation.*



(a) (b)

Figure 5: *(see color plate) The broken dam simulation with feature guidance (a) enabled, and (b) disabled.*

The feature map for the input videos was specified manually. This technique can be used to add visual complexity to a simulation and yield an effective impression of turbulence.

Another application of our technique is to perform texture transfer between real videos for special effects. We use one input video as the source of texture patches, and another target video to define the desired feature map and flow field. By performing texture synthesis restricted to the 2D grid of the target video, we can generate a video which has the appearance of one video and the temporal behavior of the other. Several results are presented in Figure 8.

Our technique is flexible enough to extend beyond synthesis of appearance only. We have experimented with synthesizing complex, dynamic surface behavior during phenomena such as boiling. A boiling simulation is performed on a small $64 \times 64 \times 20$ grid representing a slice of volume near the surface of the fluid. This yields a sequence of distance fields representing the bubbling surface. We treat this sequence as a video texture with 20 *channels* – instead of the regular three (r,g,b) – where each channel stores the distance (from the liquid surface) at a location transverse to the slice. The 2D temperature field at the surface is treated as the feature map. Using novel temperature maps as input features to our system, one can synthesize the corresponding distance fields representing boiling fluids over much larger domains. In Figure 9, we show a frame from a $320 \times 320 \times 20$ boiling sequence synthesized using our technique.

5.3. Discussion and limitations

The results demonstrate the flexibility of our technique. It can be applied to many different scenarios with only a small amount of user guidance. One limitation of our approach is that if the flow in the video is too fast and complex, then the optic flow computation can be inaccurate. This leads to increased noise in the synthesis since a lot of the motion is conveyed as texture evolution and can interfere with the flow in the target scene.

Our technique is computationally intensive, and our implementation is unoptimized and does not incorporate some very recent work for fast optimization-based texture syn-

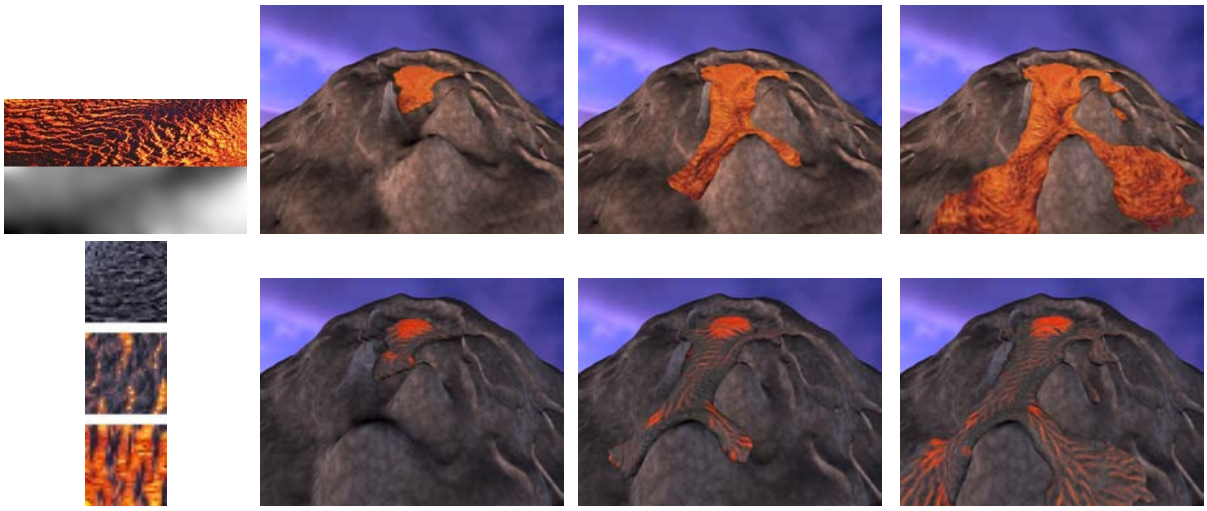


Figure 6: *(see color plate) Two different kinds of lava flowing in complex environments.*

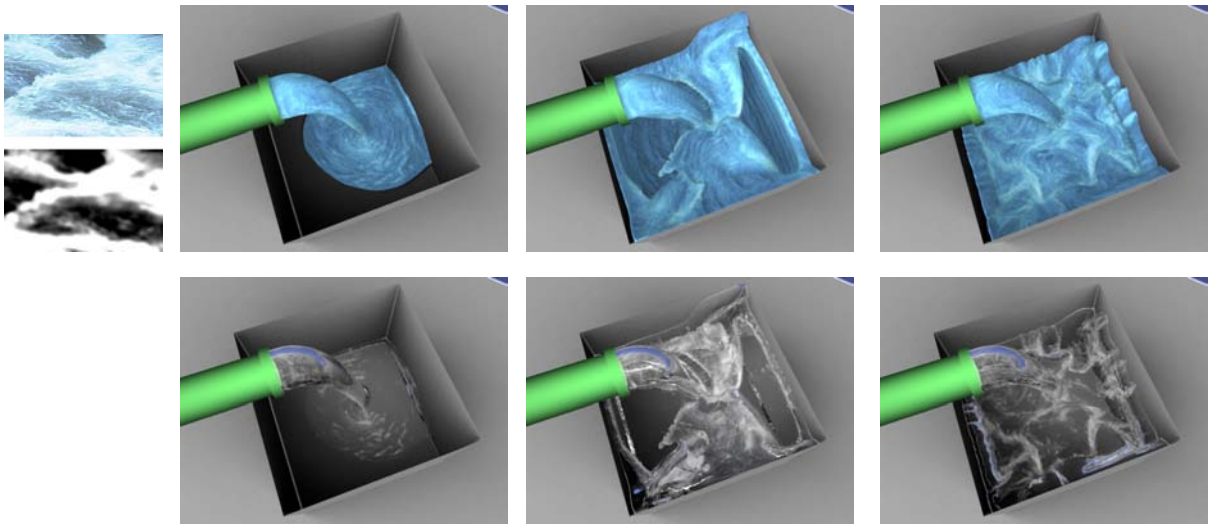


Figure 7: *(see color plate) Water gushing into a box. The input was a video of a turbulent river; one frame is shown on the left. The synthesis output can be used as a diffuse texture (upper row) or an opacity map (lower row) for different visual effects. A comparison with an untextured rendering can be seen in the accompanying video.*

thesis [HZW*06]. Depending on the simulation, it typically takes about 200–500 seconds per frame on a 3.4 GHz Pentium 4 processor. Further work could include research on techniques for the acceleration of our algorithm.

6. Conclusion

We have presented a novel technique for synthesis of physically-based textures over animated flows using image or video input. The key contributions are an innovative approach for using physical and geometric characteristics of flow transport to generate visually realistic textures on flow surfaces in computer simulation or video footage, and a general framework for synthesizing continuously varying textures using example-based texture optimization. Our technique produces visually convincing results for a wide range of phenomena with little user guidance.

Our approach can be used to easily enhance the visual realism of a liquid animation or automatically control or change the appearance of recorded natural phenomena in video. It may also be used for visualization and special effects. Our current implementation works with 2D image or video textures as exemplars. However, this framework can

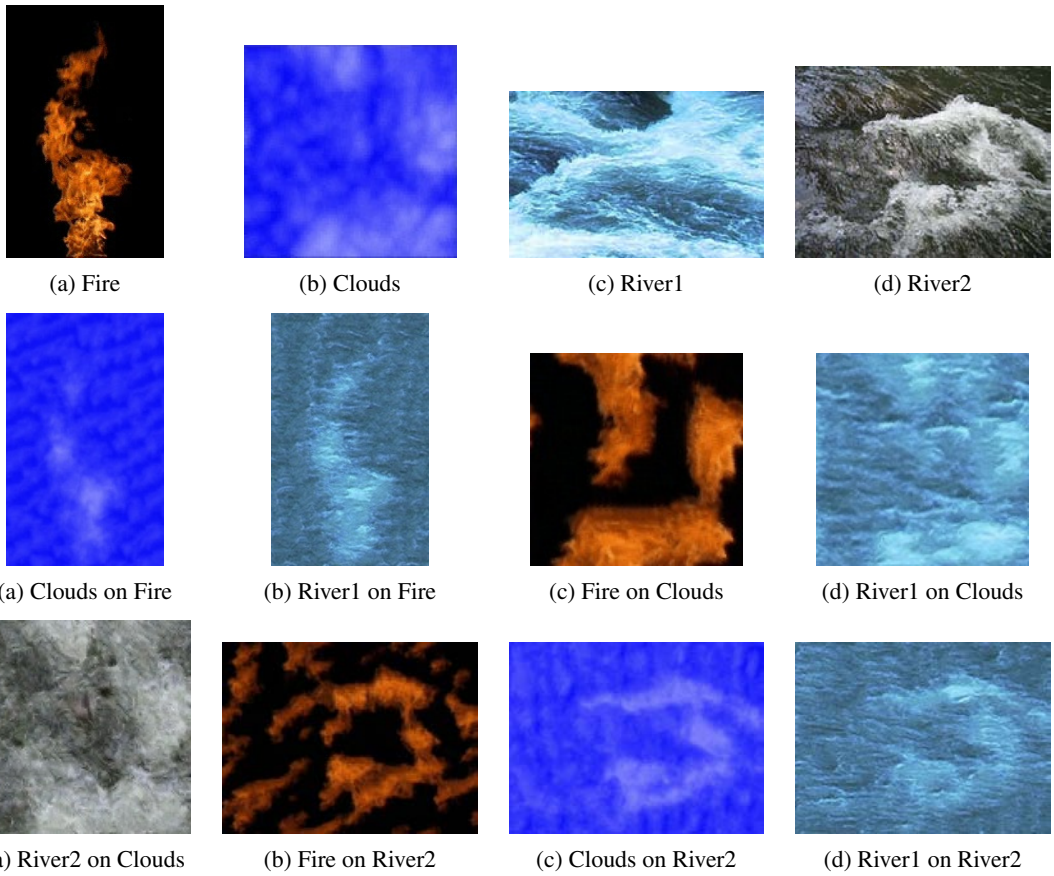


Figure 8: *Top row: Frames from the source videos. Middle and lower rows: Results of dynamic texture synthesis to generate new artistic videos using different pairs of input videos. For example, (e) is the result of applying textures from the Clouds video on the features and flow of Fire.*

be easily extended to handle 3D volumetric textures, which should open up many other possibilities, such as texture-enhanced animation of natural phenomena.

Acknowledgments

We would like to thank the reviewers for their insightful comments. This work was supported in part by Army Research Office, National Science Foundation, RDECOM, and the University of North Carolina at Chapel Hill.

References

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. In *Symposium on Interactive 3D Graphics* (2001), pp. 217–226.

[BA96] BLACK M. J., ANANDAN P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding 63*, 1 (1996), 75–104.

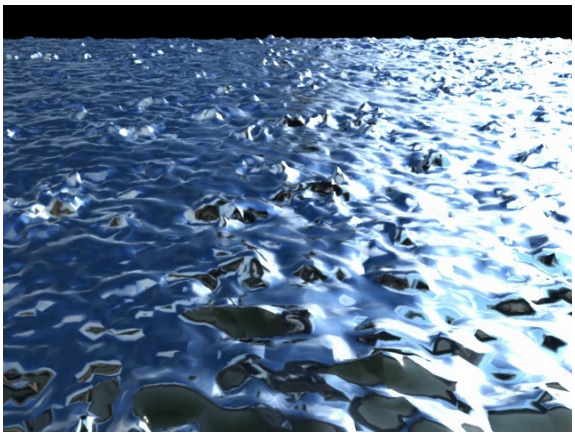


Figure 9: *Creation of complex liquid surfaces for a boiling water simulation using a texture-based representation. Shown is the surface in one frame from the sequence, obtained as the level set of the synthesized distance field. Input resolution:* $64 \times 64 \times 20$. *Output resolution:* $320 \times 320 \times 20$.

[BSHK04] BHAT K. S., SEITZ S. M., HODGINS J. K., KHOSLA P. K.: Flow-based video synthesis and editing. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3 (August 2004).

[BSM*06] BARGTEIL A. W., SIN F., MICHAELS J. E., GOKTEKIN T. G., O'BRIEN J. F.: A texture synthesis method for liquid animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Sept 2006).

[DS03] DORETTO G., SOATTO S.: Editable dynamic textures. In *IEEE Computer Vision and Pattern Recognition* (2003), pp. II: 137–142.

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *SIGGRAPH 2001 Conference Proceedings, August 12–17, 2001, Los Angeles, CA* (pub-ACM:adr, 2001), ACM, (Ed.), ACM Press, pp. 341–346.

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *ICCV* (1999), pp. 1033–1038.

[FR86] FOURNIER A., REEVES W. T.: A simple model of ocean waves. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 75–84.

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *SIGGRAPH 2001 Conference Proceedings, August 12–17, 2001, Los Angeles, CA* (2001), ACM, (Ed.), ACM Press, pp. 327–340.

[HZW*06] HAN J., ZHOU K., WEI L.-Y., GONG M., BAO H., ZHANG X., GUO B.: Fast example-based surface texture synthesis via discrete optimization. *Vis. Comput. 22*, 9 (2006), 918–925.

[KAK*07] KWATRA V., ADALSTEINSSON D., KIM T., KWATRA N., CARLSON M., LIN M.: Texturing fluids. *To Appear in IEEE Transactions on Visualization and Computer Graphics* (2007). DOI: 10.1109/TVCG.2007.1044.

[KCC*06] KIM J., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. In *Eurographics/SIGGRAPH Symposium on Computer Animation* (2006), Feneller D., Spencer S., (Eds.), Eurographics Association.

[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. *ACM Transactions on Graphics 22*, 3 (July 2003), 277–286.

[LH06] LEFEBVRE S., HOPPE H.: Appearance-space texture synthesis. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 541–548.

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[MKC*06] MERTENS T., KAUTZ J., CHEN J., BEKAERT P., DURAND F.: Texture transfer using geometry correlation. In *Proceedings of Eurographics Symposium on Rendering 2006* (2006), Akenine-Möller T., Heidrich W., (Eds.), Eurographics Association.

[Ney03] NEYRET F.: Advected textures. *Symposium on Computer Animation'03* (July 2003).

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum 22*, 4 (2003), 775–792.

[SAC*99] STORA D., AGLIATI P.-O., CANI M.-P., NEYRET F., GASCUEL J.-D.: Animating lava flows. In *Graphics Interface* (Jun 1999), pp. 203–210.

[SH81] SCHUNCK B. G., HORN B. K. P.: Constraints on optical flow computation. In *Pattern Recognition and Image Processing* (1981), pp. 205–210.

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 489–498.

[ST94] SHI J., TOMASI C.: Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 1994), pp. 593–600.

[Sta99] STAM J.: Stable fluids. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), Rockwood A., (Ed.), Addison Wesley Longman, pp. 121–128.

[TK91] TOMASI C., KANADE T.: *Detection and Tracking of Point Features*. Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, Apr. 1991.

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Siggraph 2000, Computer Graphics Proceedings* (2000), Akeley K., (Ed.), Annual Conference Series, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, pp. 479–488.

[WY04] WU Q., YU Y.: Feature matching and deformation for texture synthesis. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM Press, pp. 364–367.

[ZZV*03] ZHANG J., ZHOU K., VELHO L., GUO B., SHUM H.-Y.: Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Transactions on Graphics 22*, 3 (July 2003), 295–302.