COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2013; **24**:225–235
Published online 8 May 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1501

SPECIAL ISSUE PAPER

Fire pattern analysis and synthesis using EigenFires and motion transitions

Nima Nikfetrat* and Won-Sook Lee

School of Electrical Engineering and Computer Science (EECS), University of Ottawa, Ottawa, Canada

ABSTRACT

We introduce novel approaches of intuitive and easy-to-use realistic fire animation, starting from real-life fire by image-based techniques and statistical analysis. The results can be utilized as a pre-rendered sequence of images in video games, motion graphics, and cinematic visual effects. Instead of physics-based simulation, we employ an example-based principal component analysis and introduce "EigenFires." We visualize the main features of various fire samples to analyze their tracks and synthesize a new fire by combining various fire samples, recorded with high frame rates, in order to edit given sequences of fire animations. For this purpose, we present how to recognize similarity of the shapes of fire in order to change the pattern from one style of fire to another distinct style of fire procedurally. Our techniques require very little parameter tuning, compared with conventional physically based fire synthesis, video textures, and dynamic textures. A similar level of visually pleasing compressed fire is also easily produced by using principal component analysis techniques.

KEYWORDS

fire; analysis; synthesis; procedural; EigenFire; transition

***Correspondence**

Nima Nikfetrat, School of Electrical Engineering and Computer Science (EECS), University of Ottawa, Ottawa, Canada.
E-mail: nnikf006@uottawa.ca

1. INTRODUCTION

Fire, explosion, and smoke have been deployed in movies and motion picture industry, videos games, and TV commercials for more than two decades. Every year, a new approach has been introduced to replace such phenomena with computer-generated simulations. However, fluid is one of the most difficult examples of digital simulations. Expenses of processing and rendering fluid simulations, and difficulty of producing realistic simulation by nonexpert animators are possible reasons. Here, our approach provides a practical alternative, in which the results are reflecting the realism as generated from real videos of fire, but it still allows animators to control the fire. Our results can also be extended to 3D. Our research, as a video-based procedural technique, is more realistic looking in comparison with existing procedural approaches and is easier to control and faster in comparison with physical-based approaches.

Our fire dataset is composed of small-scale fire, recorded with a professional high-definition (HD) video camera at frame rates of 60 fps. After transforming our training fire dataset into "EigenFires," which is a way to represent distinguishable and meaningful features of various fire samples, we visualize a pattern of fire temporal movement in 3D principal component analysis (PCA) subspace. In addition, we also perform similarity analysis in PCA subspace to generate new procedural motions that do not necessarily exist in the input videos. Anyone is capable of building a new database of EigenFires with their own camera and videos, or loading one from our preprocessed library.

For this purpose, we introduce new methods of transitions and loops. Unlike video texture [1] that intends to fool the eyes for turbulent phenomena with a small number of similar frames in a single video sample, we put our effort on generating more frames stochastically as our motion transitions (e.g., 26 frames) from various fire samples. A mixture of various fire samples can also be synthesized using the main EigenFires, in more intuitive and interactive ways in comparison with dynamic textures [2,3].

2. RELATED WORK

2.1. Video-based Approaches

In video texture [1], a video is analyzed using L_2 distances, and on the basis of identification of good transition points,

a new one is synthesized. It is performed by inserting and connecting transition stochastically through a small number of frames, morphing, and blending. In a different strategy, we perform analysis and morphing techniques all in PCA subspace, and our aim is creating a new fire animation by stochastically inserting larger similar frames from various fire samples of high frame rates as our motion transitions. It means each generated motion transition is a short new procedural animation of fire by itself, without repeating the regular parts of a video.

Graph cuts algorithm [4] is an approach that can enhance transitions of video texture for turbulent scenes using two 3D spatio-temporal texture patches from optimal steams, instead of regular blending. However, their aim is to synthesize additional textures suitable for transitions, whereas our main objective is selecting similar frames from a rich fire database of various styles. Flow-based video synthesis [5] is also a way of synthesizing long videos of natural phenomena, by tacking and merging texture patches along desired lines.

Dynamic textures are presented by Doretto *et al*. [2] as a noise-driven linear dynamic system to synthesize a new series of frames from the original video. This idea is extended in Doretto and Soatto [3] by interactively modifying the temporal statistics of a video-based model of a dynamic texture to change the speed or intensity of the driving noise. Subsequently, Yuan *et al*. [6] propose a closed-loop linear dynamic system with hidden state vectors to synthesize an image. A work from Masiero and Chiuso [7] also deals with texture synthesis using nonlinear dynamical models and compares it with previous linear methods.

In our research, we present another way of synthesizing fire using EigenFires that can enhance the capabilities of dynamic textures and can be used for transitions. By considering a training set as grayscale images of various fire sequences, we can also avoid color artifacts that usually appear in synthesized images of dynamic textures because of using fewer dimensions of colorful datasets.

For fire recognition literature, an image-based technique is presented by Li *et al*. [8] for burning states recognition of poor-quality flame images from rotary kiln. Although they examine the status of low-quality flame images, the data are very limited, and visualization and analysis are not performed as we carry out. Another fire recognition algorithm is proposed by Hongliag *et al*. [9] using multi-features fusion algorithm.

2.2. Particle-based Approaches

Particle-based methods have been very popular and used in References [10–13] to simulate flames, which would give the user easy control over parameters such as external forces. Chiba *et al*. [14] employed a basic 2D particle system using a vortex field. Spherical billboard rendering [15], texture splats [16], and particle-based volume rendering [17] are examples of fire that are modeled or rendered by particle systems. However, a particle-based system may not always result in a visually appealing fire, unless it is combined with other physical and combustion methods [18,19]. In contrast, our research delivers high-quality results almost the same as captured videos, in exchange of losing some flexibility in comparison with particle-based methods.

2.3. Physically based Approaches

Despite the fact that physically based approaches [20–22] require significant amount of time to be processed, they mostly produce high-quality visualization of fire if the correct parameters are set. Nguyen *et al*. [23] use a semi-Lagrangian stable fluids approach to model both fire and smoke. A stochastic Lagrangian approach and a chemical composition evolution model are used by Adabala and Hughes [24]. A combination of fluid and combustion models was discussed by Min and Metaxas [25], and Pegoraro and Parker [26] employed a detailed simulation of the radiative emission and refractive transfer to achieve realistic renderings of fire. Simulation of gaseous phenomena in turbulent wind fields was depicted in [27] using a clustering algorithm, and the gas was modeled as a fuzzy blobby.

In our approach, defining external forces to control the fire is notably different. The idea is to record various real-life fire videos for a few minutes with many motions (e.g., wind effect). Later, the animator may mark and label corresponding ranges and connect them. The drawback is that animators might not be able to produce a desired motion if the recorded fire animations are very limited. However, a strong advantage of our nonphysical-based approach is that it supports basic users with no knowledge of dynamic fluids.

2.4. Procedural Approaches

Current procedural approaches tend more to imitate the characteristics of real flames by utilizing procedural noises and offsetting methods to incorporate turbulence into the fluid. Fuller *et al*. [28] take advantage of an improved Perlin noise and M-Noise on GPU, and then combine it with an interesting curve-based volumetric free-form deformation to create fire procedurally in real time. Although it produces a good looking fire, the animations are not close to natural fire, as it is a system solely based on random noises.

Lamorlette and Foster [29] propose a technique that flame profile is created from a set of points as spine of the flame, based on the observed statistical properties of real fire. The complexity of working with such a system is the main drawback here.

A model of fire is represented in [30] that can spread over the meshes using a skeleton technique that forms the flame. This approach is also utilizing noise functions or user-defined parameters for air velocity field to animate the skeletons of the flames, which actually reduces the realism.

DOI: 10.1002/cav

3. CAPTURING AND PREPROCESSING

For this research, we picked a variety of materials as our fuel and ignited them in an isolated space, in absolute darkness in front of a black background, and recorded videos in HD format at 60 fps. Real-life fire capturing has been limited in terms of shape and size variety because of the safety regulation. Our database is composed of three fire samples. We name those samples respectively as fires *A* (*a torch with lighter fuel*, *500 frames*), *B* (*a pack of three solid fuel cubes*, *1000 frames*), and *C* (*a single burning cube*, *697 frames*). Figure 1 shows one frame of the selected three real-life flames, 2197 images in total.

4. EIGENFIRES

Whereas the position of the cameras is fixed, we have three videos of fire with different heights and widths, created by different fuels. Therefore, we reposition and align them at the center of the fuel. After removing the background, principal components, which are represented as EigenFires, are calculated for 2197 frames of our three fire samples based on algorithms described in [31,32]. The images are cropped to 410×670 resolutions to avoid unnecessary calculations for the black background. Considering N as a dimension of eigenvectors, each image is dealt with as a single data point in $N = 274\,700$ dimensional space.

4.1. Contribution of EigenFires Using Weights

By projecting our fire images into its EigenFire, a vector of weights is constructed. Each weight represents the contribution of its EigenFire to form a fire. By assigning a negative or positive value for weights, we may reconstruct a new fire that can be a combination of three fire samples. The definition of weight (w) and projection is subtracting an image I with the average of images and then multiplying it with its EigenFire:

$$w = EigenFire^T \left(I - I_{\text{Average}}\right)$$

The weights are numbers in a range of $[-15000, +15000]$ for our samples. This range changes on the basis of the training set of images. Some of the EigenFires are showed in Figure 2.

For a better understanding, assume a blank image as I. The influence of a negative value for a specific weight is similar to filling image I with a mask using dark pixels of corresponding EigenFire visible in Figure 2. Similarly, a positive value fills image I using lighter pixels of that EigenFire. The gray regions remain almost unchanged, such as the background.

To elucidate the pattern between data points in PCA subspace, the data points of the first three principal components are visualized in 3D subspace in Figure 3. This figure shows that the data points related to each video of fires *A*, fire *B*, and fire *C* are distributed apart from each other. The center of the coordinates system is a vector of weights with zero values that produces an average image. The main characteristics of these fire examples produce a spiral motion, which corresponds to our observation in real life where a fire shrinks and expands repeatedly. This 3D subspace visualization is a very good tool to observe the temporal movement of a fire, and this pattern analysis will help natural fire synthesis. The previous work such as video and dynamic texture [2,1] do not have capacity in this direction.

Figure 1. Our raw recorded video samples. (*A*) A torch soaked in lighter fuel, (*B*) a pack of three solid fuel cubes, and (*C*) a burning cube.

5. FIRE MODELING WITH EIGENFIRES

In this section, we describe various applications of new fire modeling using existing fire examples. Assume an artist chooses two ranges of frames from any of three fire samples; for instance, he/she chooses $T1 = [800–880]$ (from fire *B*) and $T2 = [2000–2050]$ (from fire *C*) and plans to connect these separated animations of fire together ($T1 + T2$). Because the last frame of $T1$ belongs to fire *B* and the first frame of $T2$ belongs to fire *C*, then a simple attachment of two frames faces a sudden jump in transition from $T1$ to $T2$. How can we construct new frames to fill this gap? There are also other possible scenarios. What if the artist is interested in creating a loop of animation for $T1$? Or sequence of animations for "$T1 + T2 + T2 + T1$." The following two sections describe our methods for $T1 + T2$ scenario.

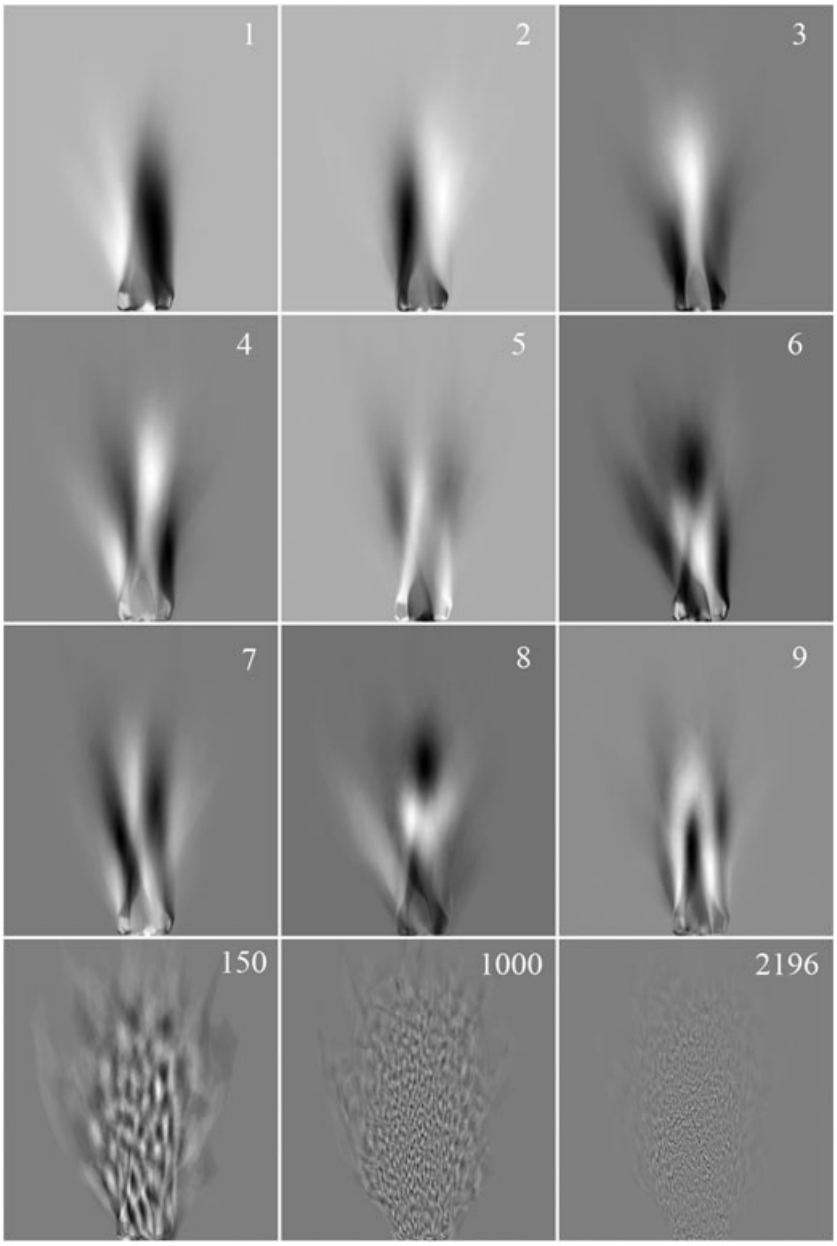

Figure 2. Principal components of numbers 1 to 9, 150, 1000, and 2196 are viewed as EigenFires.

5.1. Motion Transition with Low-pass Filter

This technique can be considered as rapid and easy to implement solution for blending/morphing in video textures, which is practical for chaotic phenomena. In this approach, the procedure to transit from one fire to another fire is averaging the weights corresponding to a few frames around the end of $T1$ and the beginning of $T2$. At first, it is required to build a new vector that is composed of vectors of weights for the images of $T1$ and $T2$, respectively. 1D low-pass filter with a kernel of $K = [1/3, 1/3, 1/3]$ can be applied on this vector to obtain the average of weights for each dimension. Now, we may reconstruct new images with this new vector of weights. Because the length of kernel was 3, the last two images of $T1$ and the first two images of $T2$ can be picked and inserted as our transition between $T1$ and $T2$. Because the first several EigenFires represent low-frequency information of fire, we only utilized the first 50 EigenFires in the process of averaging, which are only 0.02% of the total number of principal components. Thus, artifacts are avoided that usually appear as sharp layers of target images. The result is a morphing between our target images for two distinct shapes of flames.

DOI: 10.1002/cav

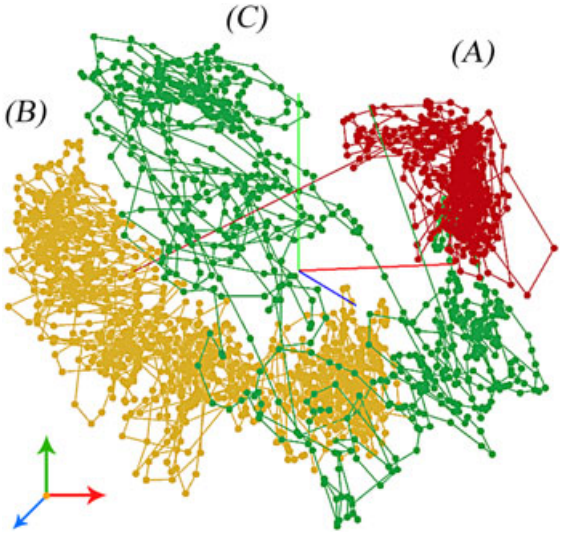

Figure 3. All images projected in 3D principal component analysis (PCA) subspace, illustrated as a series of weights for the first three EigenFires (red, green, and blue axes, respectively). Points are labeled with their relative fire name. The connected lines show the relation of consecutive frames of fire that allows us to track the changes over time in 3D PCA subspace.

Other kernels such as $K = [1/5, 1/5, 3/5]$ may also be chosen for the end part of $T1$ and then flipping it for the beginning of $T2$ to create a different deformation by giving more weights to our target images. Figure 4 shows three reconstructed frames to fill the gap between $T1$ and $T2$. To include fine details, we may copy a few weights randomly from our target images to this average vector, similar to the behavior of dynamic textures. User's control on weights, as a way of synthesizing fire, can also deform and improve the outputs of this technique.

5.2. Motion Transition with Recognition

In this section, we suggest a few methods based on identification of similarities to model fire. The approach is filling the gap between our $T1$ and $T2$ by choosing almost similar images to finalize our transition. Let the last frame of $T1$ be image α and the first frame of $T2$ be image β. Thus, α_j and β_j describe the jth similar images to initial images α and β. The purpose is acquiring one frame in each step that is not exactly the same but can be selected as our next frame. Because finding very similar shapes produces unrealistic animations and might be displayed as slow motion, defining a minimum threshold θ_{min} is necessary (Figure 5). In our transition methods, we only set a single θ_{min} for each transition based on the fire samples that are going to be combined and how much changes we permit to occur.

We propose two techniques of recognition, "Direct Bridging" and "End Growing Connection." In both cases, the animator chooses a specific number of steps, N, in order to find at most N numbers of similar images to be inserted between $T1$ and $T2$.

5.2.1. Direct Bridging Method

In Direct Bridging Method, firstly images of our fire videos are transformed into EigenFire space if they are not already included in the database. Next, on the basis of the value of steps, the multidimensional line between two corresponding points of images α and β in EigenFire space is divided equally by N. In other words, we insert N virtual new data points by dividing values of weights for E

Figure 5. Three different frames from our fire videos that are recognized similar to each other by finding the shortest Euclidean distances in principal component analysis space. By calculating thresholds for our fire videos, we can avoid slow motions or fast forwards in animations and still generate larger motion transitions.

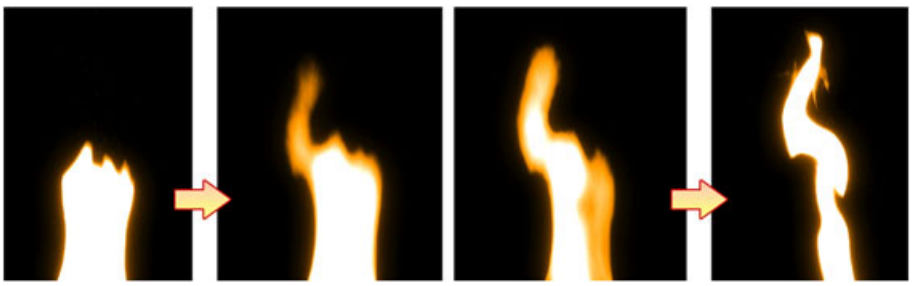

Figure 4. Low-pass filter approach. (left) Last frame of $T1$ and (right) first frame of $T2$ (middle images) two reconstructed images using 50 EigenFires that appear as morphing. A synthesized color and bloom effect is applied to all images.

dimensions. Reconstructing images with these new data points, which are N new vectors of weights, produce unpleasant layers of flames. Therefore, the solution is finding the closest existing data point to each of N points in each step by minimizing the Euclidean distance between them.

The advantages of this method are the high speed of identifications and the flexibility to define the exact number of steps for transitions. The disadvantage here is the possibility of recognizing images that are closest in distance for one step but are not close enough to the identified frame of previous step. This might reduce smoothness of motions without human verification if a large N is chosen for steps. The second row of Figure 6 shows the concept of this method. A decent example of this technique is presented in Figure 7 that allows animators to quickly build loops for animations by finding a certain number of similar images.

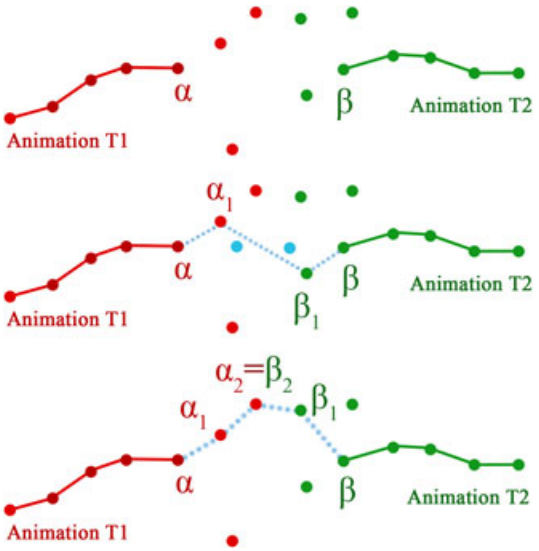

Figure 6. (Top) The concept of our recognition techniques, visualized in 2D. We connect α and β through finding a few close data points in EigenFire space to obtain a smooth transition. (middle) In Direct Bridging method for $N = 2$ steps, we first insert two temporary data points (in blue) between α and β, and then look for the nearest points to each of them. (bottom) End Growing Connection Method for maximum $N = 20$ steps, the blue dotted line is our path using the algorithm of Table I, and it stops in three steps by finding a shared image.

5.2.2. End Growing Connection Method

In End Growing Connection method, the animator chooses a large N as the maximum possible number of steps and then looks for a pattern that gradually connects image α to β through finding similar images in between, which are stored as α_j and β_j. The algorithm stops after reaching a point nearest to both α_j and β_j. In other words, we are building a path that can connect maximum N data points in E dimensions. Remember that each data point is representing one image in PCA subspace. Reconstructing images of this path should generate a short animation to connect $T1$ and $T2$ animations. In our approach, we try to detect maximum $N/2$ similar images for α and $N/2$ for β until we meet one of the following possibilities:

(1) A shared image, similar to both α and β, is found: $\alpha_j = \beta_k$.
(2) No shared image is detected, but the distance between α_j and β_j is small and less than θ_{min}.
(3) All identified images are bigger than θ_{min} after N steps.

In the first two cases, the proper path or pattern is constructed, and the algorithm stops successfully. In the last case, we should choose a larger step or employ the previous Direct Bridging method or morphing with low-lass filter.

Table I summarizes a simple algorithm to find a path that is constructed of short distances less than the difference between α and β, and above our threshold. In each step, we aim for a new image even nearer to the other side, in which the other side is α or β. The path is also optimized by removing some images from α_j and β_k lists at the end, if we encounter dramatic changes of distances. The third row of Figure 6 illustrates the concept of this method.

This algorithm yields very smooth transitions, particularly for one style of fire. The drawbacks are the unknown numbers of steps and failure for very distinct

Figure 7. An example of Direct Bridging method to connect $T1$ and $T2$ animations to simulate a loop of our animations. (left) Image α, (right) image β, and (middle) three images that are found similar by $\theta_{min} = 50$ in $N = 3$ steps. Reconstructed images are colorized by extracting colors from original videos, and finally, a bloom effect is applied.

DOI: 10.1002/cav

Table I. An algorithm for End Growing Connection method to find an optimized short path suitable for procedural fire animation.

```
For steps 1 to N/2 do   /* find 2 images in each step */
   foundStep = false
   α_{j=[1,..,2197]} ← Sorted distances of new α to all images
   β_{k=[1,..,2197]} ← Sorted distances of new β to all images
    j ← 0, k ← 0
   While exists items in α_j and β_k, and not foundStep do
      If (any element of α_j = any element of β_k) Then
         Return   /* A shared image is found */

      /* Compute distances in E dimensions: */
      D ← ||α - β|| ,   D12 ← ||α_j - β_k||
      D1 ← ||α - β_k|| , D2 ← ||β - α_j||
      shortToα ← false,  shortToβ ← false

      If (D - D12 < θmin) Then
         If (D1 – D ≤ θmin) Then   shortToα ← true
         If (D2 – D ≤ θmin) Then   shortToβ ← true

         If (shortToα and shortToβ)
            j ← j + 1,    k ← k + 1
         Else
            If (Not shortToα) Then     k ← k + 1
            If (Not shortToβ)  Then    j ← j + 1
      Else
         Remove_duplicates()
         α ← α_j ,     β ← β_k,  foundStep = true
   End While
Optimize_Path()
```

flames with a small dataset. Simulation of an external force (wind) is illustrated in Figure 8 using the End Growing Connection method. Figure 9 shows the interesting results of generating a new fire animation that is not recorded by cameras. Note that we only set a single θ_{min} for each transition to keep the amount of changes stable for the entire steps.

In both of the recognition methods, an image is also rejected if it is already identified in previous steps or if it is one of the last few frames of the video before α and after β. This way, we avoid deadlocks or moving backward along the same pattern, and we are still following a procedural approach that does not repeat parts of the video by means of higher frame rates, as we find larger number of images for transitions, different from video textures.

5.3. Fire Synthesis with EigenFires and Weights

Considering the fact that motion transition with low-pass filter might not always lead to the expected result, manual weight adjustment can produce additional details that an

Figure 8. An example of End Growing Connection method to simulate an external force (wind). Recognition of similar flames is restricted to fire *C* only. The end of *T*1 is a flame with a straight shape (left), and the beginning of *T*2 is a flame with a slope (right). For $\theta_{min} = 1500$, the algorithm stops after 19 steps.

Figure 9. An example of End Growing Connection method to simulate starting a fire. *T1* is only a single frame from fire *A*, which is also our image α (left), and *T2* is a regular animation of flame from fire *C*, and β is the first frame of *T2* (right). This example demonstrates that with some creativity, animators can produce new animations, which are not yet recorded. For $\theta_{min} = 400$, the algorithm stops after 14 steps.

animator is looking for. After projecting a specific image into a PCA subspace, the weights of certain EigenFires can be modified easily via user interface, and new fire is displayed in real time. Because our first 50 EigenFires carry major features of flames, it is not even very tricky to start from scratch. We first assign zero values to weights of all dimensions and then allocate new values by taking a look at the generated EigenFires of database. Figure 10 illustrates an example with a manipulation of only eight EigenFires (0.003% of the full dimensions!).

Assigning small weights to any EigenFire larger than 50 generates some fine details, having similar effect to applying procedural noises. The drawback of manual weight adjustment is the difficulty of creating realistic animations. As a result, we suggest combining this method with previously discussed motion transitions to follow the patterns of real flames, similar to dynamic textures but modifying the fire more intuitively using the main features of visualized EigenFires.

6. RESULTS

We ran our experiments on a system with Intel Core i7-2600 k 3.4-GHz CPU and 8 GB of memory. We developed our application with C++ language and cross-platform Qt framework for user interface. OpenGL 3.3 is also employed for 3D visualizations, and our 64-bit implementation allows users to process very large video files.

For 2197 fire images of 410×670 pixels, it takes around 1 hour to calculate all the stages of generating EigenFires including loading data, preprocessing, and PCA calculations. More precisely, PCA calculation takes about 45 minutes, and projecting all training images into PCA subspace takes around 20 minutes. However, these calculations need to be performed only once to prepare the initial database. The PCA calculation depends on data size greatly. For example, the PCA steps take only 12 minutes for a dataset of 1000 images. Except initial PCA calculations and database preparation, all other techniques are spontaneous, meaning they run at real-time interactive rate.

The optimized number of $E = 1100$ EigenFires (50%) can be considered for both high-quality reconstructions and proper detections. A smaller number, around $E = 700$ EigenFires, is also sufficient if the purpose is only reconstruction rather than recognition, because the images only lose some noises inside the boundaries of the flames. Figure 11 illustrates this procedure. Note that we may set two parameters as E: one for accuracy of recognition and another one for quality of reconstruction. Table II shows a summary of the performance, excluding reconstruction times because what we usually need is a list of frames as our transitions. In the last three figures of this paper, our recognition steps are performed with 1100 EigenFires. As it is obvious from the table, we might recognize one or a few more similar images by using fewer numbers of EigenFires, due to the slight change in accuracy.

There are a few limitations in our system. Although End Growing Connection method performs better than our other

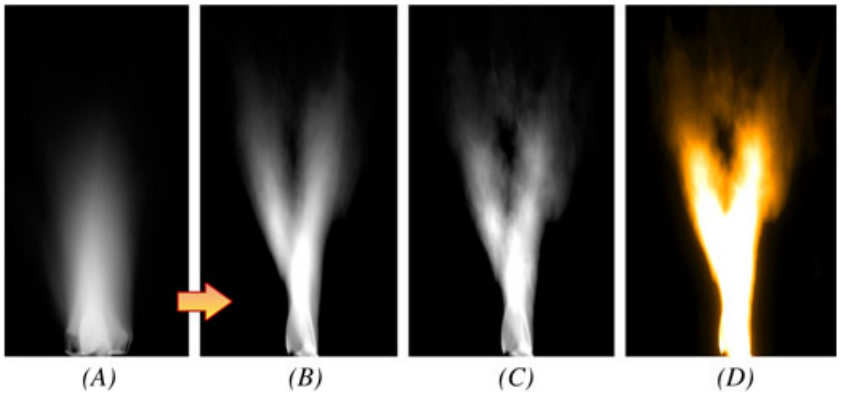

Figure 10. (*A*) The average image; different weights deforms this shape. (*B*) Weights assigned to EigenFires of 2, 3, 5, 6, and 7 for global shape. (*C*) Assigning more weights to EigenFires of 50, 80, and 500 for distortions and some details. (*D*) The result with synthetic coloring.

DOI: 10.1002/cav

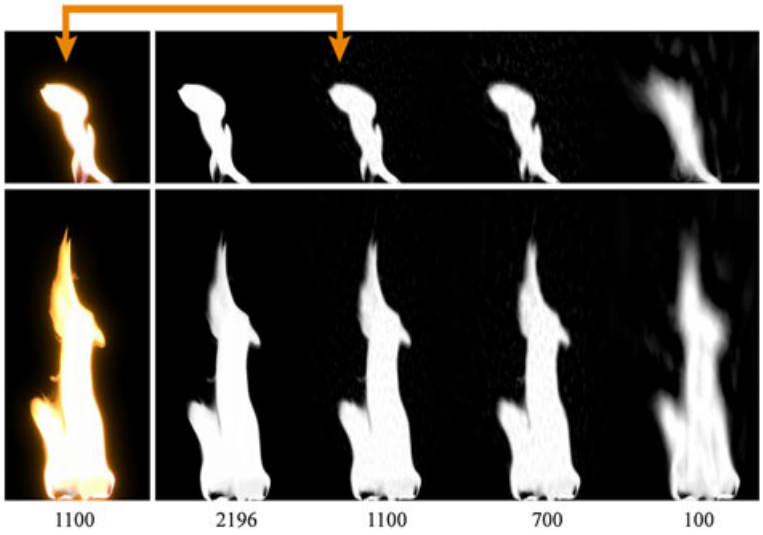

Figure 11. Reconstruction of fire samples with different numbers of EigenFires, as written below each column.

Table II. Performance statistics.

Technique	Fire	EigenFires	Total time (milliseconds)	Steps	Time per step (milliseconds)
DirectBridging(Figure 7)	*A, B, C*	2196	101	3	34
		1100	65	3	22
End Growing	*C*	2196	169	18	9
Connection (Figure 8)		1100	104	19	5
End Growing	*A, B, C*	2196	416	13	32
Connection (Figure 9)		1100	294	14	21

motion transitions for various fire samples, it might fail to find the best shared image if the chosen dataset is small or includes limited motions. The initial calculation time of PCA might also be high for very large lengths of HD videos. However, it can be significantly reduced by resizing the training images and then referring to the corresponding frames of the original HD video for the final animation.

The color of fire can also be extracted from original images using reconstructed images as our masks, or it can be generated synthetically by creating an image in HLS color space (color components of hue, lightness and saturation) with the luminance of our gray images and the hue and saturation of another colorful image (e.g., solid orange). To produce a basic glow effect, the trick is duplicating our tinted fire image, applying several passes of Gaussian smoothing with a large radius, and finally blending that blurred image with our colorized image.

7. DISCUSSION FOR 3D EXTENSION

Image-based techniques can be employed to reconstruct 3D fire by utilizing our current 2D fire modeling results. Hasinoff and Kutulakos [33][34] present flame sheets and

Figure 12. Integration of our outputs into Autodesk 3D Studio Max using camera-facing billboards.

density sheet decomposition for reconstruction of semi-transparent scenes and flames, using interpolating views, applicable to our final results. A sparse view tomographic approach may also be applied for reconstructing a volumetric model from multiple images by placing more cameras around the fire [35]. Therefore, one view or a maximum of two views can be used for the recognition step to generate new frames between two handpicked motions. Krüger and Westermann [17] perform physics-based 2D simulation for two slices and extrude it to 3D, using particles and textures. The difference is that our 2D simulation is a procedural approach of two camera views. Figure 12 shows our results in 3D using classical camera-facing billboards.

8. CONCLUSIONS AND FUTURE WORK

We explored fire behavior using PCA and introduced new transition approaches of modeling fire from a combination of fire samples by looking for similar shapes of flames and expanding the pattern of fire. This proposed method provides an intuitive fire modeling system where a naive user can easily design fire animation with a story in mind without knowing any physical parameters.

Our methods can be considered as an extension of the capabilities of both video and dynamic textures focusing on fire animation. However, our PCA approach, different from existing methods, is capable of visualizing 3D subspace points to understand how different fires behave, and the tracking of points in PCA space provides great potential for further fire analysis and modeling. Figure 3 shows different spiral patterns for each fire *A*, *B*, and *C*, which reveals the great potential to build synthetic but very natural fire movement. By decreasing the PCA dimension and synthetic coloring, we can compress the fire database up to 78% of the entire size by choosing only the most important EigenFires.

Future work aims to increase the database to cover the variety of fires, including large-scale fire, and to improve the user interface to be more intuitive in a touch-based environment. Accordingly, more accurate results can be obtained from high-speed cameras in full HD that provide frame rates of around 300 fps.

REFERENCES

1. Schödl A, Szeliski R, Salesin DH, Essa I. Video textures. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH'2000), 2000; pp. 489–498.
2. Doretto G, Chiuso A, Soatto S, Wu, Y. Dynamic textures. *International Journal of Computer Vision* 2003; **51**(2): 91–109.
3. Doretto G, Soatto S. Editable dynamic textures. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 2003; 137–142.
4. Kwatra V, Schödl A, Essa I, Turk G, Bobick A. Graphcut textures: image and video synthesis using graph cuts. In ACM SIGGRAPH Papers, New York, USA, 2003; 277–286.
5. Bhat KS, Seitz SM, Hodgins JK, Khosla PK. Flow-based video synthesis and editing. In ACM SIGGRAPH 2004, New York, USA, 2004; pp. 360–363.
6. Yuan L, Wen F, Liu C, Shum HY. Synthesizing dynamic texture with closed-loop linear dynamic system. European Conference on Computer Vision (ECCV), 2004; pp. 603–616.
7. Masiero A, Chiuso A. Non linear temporal textures synthesis: a Monte Carlo approach. In Proceedings of the 9th European conference on Computer Vision, 2006; pp. 283–294.
8. Li W, Mao K, Zhou X, Chai T, Zhang H. Eigen-flame image-based robust recognition of burning states for sintering process control of rotary kiln. In Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference. CDC/CCC, 2009; pp. 398–403.
9. Hongliang L, Qing L, Sun'an W. A novel fire recognition algorithm based on flame's Multi-features Fusion. International Conference on Computer Communication and Informatics, 2012; pp. 1–6.
10. Perry CH, Picard RW. Synthesizing flames and their spreading. In Proceedings of the Fifth Eurographics Workshop on Animation and Simulation, 1994; pp. 1–14.
11. Reeves WT. Particle systems-a technique for modeling a class of fuzzy objects. In Proc. of ACM SIGGRAPH '83, 1983; pp. 359–375.
12. Stam J, Fiume E. Depicting fire and other gaseous phenomena using diffusion processes. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH '95), ACM, New York, NY, USA, 1995; pp. 129–136.
13. Takahashi J, Takahashi H, Chiba N. Image synthesis of flickering scenes including simulated flames. *IEICE Transactions on Information and Systems* 1997; **80**(11): 1102–1108.
14. Chiba N, Muraoka K, Takahashi H, Miura M. Two-dimensional visual simulation of flames, smoke and the spread of fire. *The Journal of Visualization and Computer Animation* 1994; **5**: 37–53.
15. Umenhoffer T, Szirmay-Kalos L, Szijártó G. Spherical billboards and their application to rendering explosions. In Proceedings of Graphics Interface, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 2006; pp. 57–63.

DOI: 10.1002/cav

16. Wei X, Li W, Mueller K, Kaufman A. Simulating fire with texture splats. In Proceedings of the conference on Visualization, IEEE Computer Society, 2002; pp. 227–235.
17. Krüger J, Westermann R. GPU simulation and rendering of volumetric effects for computer games and virtual environments. In Proceedings Eurographics, 2005; pp. 685–693.
18. Feldman BE, O'Brien JF, Arikan O. Animating suspended particle explosions. In ACM SIGGRAPH 2003, New York, NY, USA, 2003; pp. 708–715.
19. Rasmussen N, Nguyen DQ, Geiger W, Fedkiw R. Smoke simulation for large scale phenomena. In ACM SIGGRAPH 2003 Papers, New York, USA, 2003; pp. 703–707.
20. Stam J. Real-time fluid dynamics for games. In Proceedings of the Game Developer Conference. 2003.
21. Stam J. Stable fluids. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99). ACM Press/Addison-Wesley Publishing Co., 1999; pp. 121–128.
22. Melek Z, Keyser J. Interactive simulation of fire. In Proceedings of the 10th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society, Washington, DC, USA, 2002; pp. 431–432.
23. Nguyen DQ, Fedkiw R, Jensen HW. Physically based modeling and animation of fire. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, (SIGGRAPH '02), New York, NY, USA, 2002; pp. 721–728.
24. Adabala N, Hughes C. A parametric model for real-time flickering fire. In Proceedings of Computer Animation and Social Agents (CASA). 2004.
25. Min K, Metaxas D. A combustion-based technique for fire animation and visualization. *The Visual Computer* 2007; **23**: 679–687.
26. Pegoraro V, Parker SG. Physically-based realistic fire rendering. In Eurographics Workshop on Natural Phenomena, 2006; pp. 237–244.
27. Stam J, Fiume E. Turbulent wind fields for gaseous phenomena. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, (SIGGRAPH '93), ACM, 1993; pp. 369–376.
28. Fuller AR, Krishnan H, Mahrous K, Hamann B, Joy KI. Real-time procedural volumetric fire. In Proceedings of the 2007 symposium on Interactive 3D graphics and games, ACM, New York, 2007; pp. 175–180.
29. Lamorlette A, Foster N. Structural modeling of flames for a production environment. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM SIGGRAPH 2002, New York, NY, USA, 2002; pp. 729–735.
30. Beaudoin P, Paquet S, Poulin P. Realistic and controllable fire simulation. Graphics interface 2001, Canadian Information Processing Society, Toronto, 2001; pp. 159–166.
31. Turk M, Pentland A. Eigenfaces for recognition. *Journal of Cognitive Neuroscience* 1991; **3**(1): 71–86.
32. Nikfetrat N, Lee W-S. Fire visualization using eigenfires. Intelligent Computer Graphics 2012, Vol, **441**, Springer: Berlin/Heidelberg, 2013; 189–207.
33. Hasinoff SW, Kutulakos KN. Photo-consistent 3D fire by flame-sheet decomposition. In Proceedings of the Ninth IEEE International Conference on Computer Vision, vol. 2, 2003; pp. 1184–1191.
34. Hasinoff S, Kutulakos K. Photo-consistent reconstruction of semitransparent scenes by density-sheet decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2007; **29**(5): 870–885.
35. Ihrke I, Magnor M. Image-based tomographic reconstruction of flames. In Proceedings of Eurographics symposium on Computer animation, ACM SIGGRAPH, 2004; pp. 367–375.

AUTHORS' BIOGRAPHIES:

Nima Nikfetrat He received his master's degree in computer science from the University of Ottawa, Canada. He is currently working in Exocortex Technologies, a company that provides cutting-edge 3D tools and technologies for leading studios around the world. His experience and research interests are computer graphics, animation, video game development, and image processing.

Won-Sook Lee She is an associate professor in the School of Electrical Engineering and Computer Science, Faculty of Engineering, University of Ottawa, Canada. She obtained her degrees in POSTECH (South Korea), NUS (Singapore), EPFL, and University of Geneva (Switzerland) and previously worked in Korea Telecom (South Korea), Eyematic Interfaces, Inc. (USA) and Samsung (South Korea). Her research interests are in the area of computer graphics and animation, including human modeling and animation, haptic interface, and medical imaging.