Fire Visualization using Eigenfires

Nima Nikfetrat, Won-Sook Lee

School of Electrical Engineering and Computer Science (EECS)
University of Ottawa
Ottawa, Canada
nnikf006@uottawa.ca, wslee@uottawa.ca

Abstract. Procedural modeling of fire has been very practical and popular, but most of them are based on random parameters for the purpose of creating realistic looking flames, while physical based modeling is closer to the realism, but suffered by complicated algorithms and heavy computational requirement. Our new approach on fire does not use any physical parameters, but uses real-life fire images and applies image-based methods and statistical analysis. We visualize the shape and motion of fire to analyze them, which can be used a simple and realistic fire modeling. We employ principal component analysis (PCA) and take it to a new level by introducing "eigenfires", which are eigenvectors of the covariance matrix of fire videos, from variety of high-definition videos of real fire to visualize and understand the track of fire movement and how different flames are located in various locations. Our system provides flexibility for the artists to manipulate the ordinary style of a flame and change it to another distinct shape using a series of weights that are assigned to each eigenfire. Our method is also efficient in terms of compact representation of fire motion as PCA allows compression by cutting high dimension data for almost the same quality of the video.

Keywords: Fire, Flame, Smoke, Fluid, Procedural, PCA, Eigenface, Eigenfire, Reconstruction.

1 Introduction

Fire is an important item for various computer generated scenes. A popular way is a physical based fire modeling and animation using physical equations and parameters, which we call as microscopic approach. This approach pays attention to the interaction among nearby particles and as a result, the fire is generated. However in naked eyes, the fire has certain degree of regulation and it has not been studied well yet. In this paper we employ image-based techniques to analyze global shape of real-life fire and its motion.

Our method differs from conventional procedural fire modeling approaches. Although procedural methods might not always be highly realistic, they offer faster, efficient and economical features. The purpose of our research is introducing a technique that is procedural, but the results are more realistic-looking in comparison to available procedural approaches.

Our primary idea of employing principal component analysis (PCA) for fire is inspired by previous works from Pengcheng Xi et al. [22][23] on the topics of facial expression, in which PCA was utilized to create new facial expression, other than the common use of PCA for recognitions [20][5]. Unlike the methods mentioned above, we do not put our effort on interpolations among feature points or contours, but on image-based reconstruction techniques. This advantage is because of our large database of images consists of thousands of frames, recorded from real small-scale flames with two professional HD video cameras at frame rates of 60 fps. Fig. 1 shows this setup.

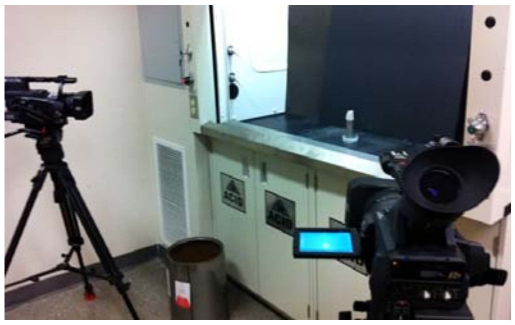

Fig. 1 The photograph of our setup for capturing small-scale flames

After transforming our training set of fire images into “*eigenfires*”, which are the eigenvectors of the covariance matrix of our set of fire images, we perform different analysis and observations to determine appropriate PCA approach for the visualization of fire tracks. Then, we make them available to animators via a user-friendly interface. Anyone is capable of building a new database of eigenfires with their own camera and video, or loading one from our pre-processed library. Unlike eigenfaces [20], which are principal components of the training set of face images and are displayed as ghostly faces, our eigenfires are exhibiting remarkable features of fire after isolating the shape of flames by removing the background; Some features such as shape, volume, holes, length, direction, and even deformation of a single flame to small pieces of flames.

Briefly, the following features outline the main capabilities of our system:

- Combining several clips (ranges) of different fire videos as training set, and building a database of eigenfires
- Background and noise removal, and image quality enhancement for PCA
- Pose and direction normalization using fitting line algorithm
- User’s control on contribution of each eigenfire by weight adjustment

In this paper, we present our results as two-dimensional sequence of images, and we only work on one view of videos even though Fig. 1 shows a setting with two cameras. The setting shown in Fig. 1 is aimed for 3D fire visualization, which is not discussed in this paper.

2 Related Work

Physically-based approaches:

Despite the fact that physically-based approaches [16][17][15][12] require significant amount of time to be processed, they mostly produce high quality visualization of fire and are more realistic if the correct parameters are set by the users for the variables of dynamic fluid equations in the system. Nguyen et al. [14] use a semi-Lagrangian stable fluids approach to model both fire and smoke; one set of incompressible flow equations to model the fuel and another one for the hot gaseous products. Thermal buoyancy is an important part of their model that influences fluid velocity. In this model, the temperature rises up until reaching a specific degree to ignite the fuel. An implicit surface, which is called blue core region, is created to divide the regions between gaseous fuel and soot. A realistic color and rendering using a stochastic ray marching algorithm complete their model. A stochastic Lagrangian approach and a chemical composition evolution model are used by Adabala et al. [1]. A combination of fluid and combustion models was discussed by Min and Metaxas [13], and Pegoraro and Parker [15] employed 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 [18] using a clustering algorithm, and the gas was modeled as a fuzzy blobby, in which advection term was responsible for moving a blob, and diffusion term to deform it by an advection-diffusion equation. Detonation shock dynamics (DSD) was used in a work by Hong et al. [9] to produce cellular patterns in flames. The fire is generated by coupling the third order DSD equations to the Navier-Stokes equations.

Producing 3D high resolution flames was achieved by Horvath and Geiger [10] on GPU for VFX of movies. The first stage is a coarse particle grid simulation which would allow the users to direct and control the motion of fire. Next, fine details will be added in the refinement stage, consisting of specific number of camera-facing image planes. Attributes of particles will be projected onto these planes. A GPU-based volume rendering and a farm of 10 GPUs make it feasible to obtain high quality results, which might not be affordable for a small business, or home users. Another drawback of this technique is the huge amount of processing time. Harris [8] describes mathematical background and implementation of 2D fluid simulations using Navier-Stokes equations for incompressible flow on the GPU, which significantly increased the performance in comparison to Stam's simulation on CPU [17][16]. Valuable GPU rendering tricks and implementations about dynamic fluids and fire are also discussed in Crane et al. [4], which is a useful reference for everyone.

An advantage of our non-physical-based approach is that it supports basic users with no prior knowledge and experience of dynamic fluids or its technical terms, such as buoyancy, vorticity and viscosity, to produce an elementary fire.

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. [7] take advantage of the improved Perlin noise and M-Noise, and then combine them with an interesting curve-based volumetric free-form deformation to create fire procedurally. Their 3D hardware-accelerated volumetric rendering allows an artist to easily manipulate and deform the fire along a curve in real-time. Although the system produces a good looking fire, the animations are not close to natural fire, as it is a system solely based on random noises. However, we provide a similar noise feature as an additional option, but using an innovative idea which is user’s control on contribution of each eigenfire via weight adjustment.

Vanzine and Vrajitoru [21] integrate the same system as above into a 3D game engine, and discuss statistical results of performance obtained from different procedural noises with variety of volume sizes, using both DirectX and OpenGL.

Lamorlette and Foster [11] propose a technique that a flame profile is created from a set of points as spine of the flame, and it is based on observed statistical properties of real fire. This curve, or spine, can break and evolve with a combination of physics-based and procedural fields, and it deforms implicit surface of the flame. Particles are sampled on this surface, in addition to applying animated procedural noises, and then they can be rendered volumetrically. The complexity of working with such a system is the drawback here, in which an animator becomes productive within a week, while it takes less than 10 minutes to learn how to work with our system.

Beaudoin et al. [3] represent fire as a small set of flames instead of working with large numbers of particles, and it can spread progressively over the meshes. Skeleton technique, which is a small group of connected particles, forms the flame animation and is moved by turbulent as a time-varying vector field. Flames are modeled using implicit surfaces and are obtained from these skeletons. Finally, the model is rendered using a ray tracing algorithm. This approach is also utilizing noise functions or user defined parameters for air velocity field to animate the skeletons of the flames, in which they actually reduce the realism. In our approach, we extract the natural patterns of fire instead of noise functions.

Amarasinghe and Parberry [2] discuss real-time rendering and deformation of burning objects on GPU, while generating procedural fire using particles. Although the deformation of burning objects looks great, the low quality of generated fire makes it unbelievable as a natural phenomenon. Therefore, we considerably pay attention to the quality criteria and recommend proper numbers of eigenfires to prevent lowering the quality.

3 Pre-Processing of Real-Life Fire Images

In our approach, we are not concerned with measuring physical properties of fire, but extracting global motions of real fire. Our large fire database includes variety of styles and motions we intentionally created and recorded.

3.1 Recording Fire Video

We captured our videos in uncompressed format for more accuracy. They sequentially form variety of motions which can be analyzed in PCA space, in order to reconstruct new copies that are following a similar pattern when we put them together. We picked variety of materials as our fuel and ignited them in an isolated space, in absolute darkness in front of a black background. Although we collected 8 varied video clips of fire, and more than 15,000 frames in total at 60fps, we chose the best three fire samples that we found suitable for this paper. We name those samples respectively as fire *A* (a torch with lighter fuel), *B* (a pack of three solid fuel cubes) and *C* (a single burning cube) throughout this paper. And construct our results based on them. Fig. 2 shows one frame of selected three real-life flames without applying our background removal step. Videos *A*, *B* and *C* contain 500, 1500 and 700 frames respectively in our experiments, which are comprised of 2700 images in total. The videos will be cropped aligned at the center of their fuel or the burning object, so that our flames are centered, and a smooth morphing can be performed later among them while reconstructing new images. Because of our restricted setup and safety regulations, we could not place our fuel far from the background. Therefore, the brightness of flames illuminated our black background in some frames, and we will eliminate it by darkening process in section 3.3.

Characteristics of our selected videos are as follows: Video *A* is a short flame with a small volume, repeatedly bended toward the left side of the view, and contains mixtures of colors, such as blue and red around the fuel. Flame *B* consists of larger width and volume, in addition to a unique shape, in which constantly separates into smaller flames or branches every couple of frames. This feature is slightly visible in Fig. 2 at the tip of flame. The third sample, fire *C*, includes a higher length, smaller width in comparison to the second sample, and more distortions due to external forces (e.g. wind) which are made intentionally during recording.

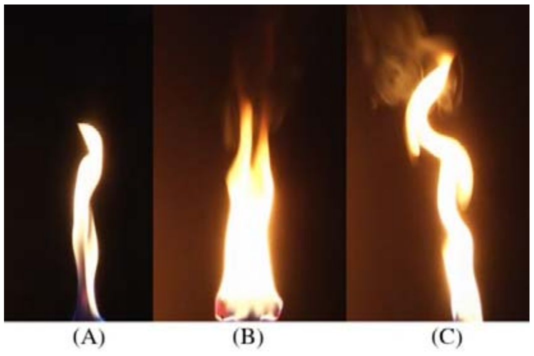

Fig. 2 Original recorded video samples chosen for this paper, cropped in this figure. (*A*) A small torch soaked in lighter fuel, (*B*) a pack of three solid fuel cubes, and (*C*) a single burning cube

3.2 Contour Detection

The first step for preparation of images is detecting contour of the flames, which is for the purpose of background removal, pose normalization, and detection of different color regions. There are many techniques that can be used to extract information about the boundaries of different objects from images. Standard snake, gradient vector flow snake, contracting curve density algorithm (CCD) are a few methods of shape matching to approximate the object's contour. However, based on our previous experiments and observations, we found them time-consuming, CPU intensive, and not proper for moving shapes, such as fluids. Utilizing these methods make the process of the detection non real-time or tedious for thousands of images of videos in HD format, and different results might be received with slight changes for consecutive frames of the same fire, due to minor errors in detections. Because of high intensity and brightness of flames in comparison to our darker background, we figured out working on range of colors directs us to obtain proper contours which are more consistent throughout the entire video while the shape of our flames changes constantly.

Finding specific range of colors is not easy in RGB color space. For example, we can detect how much of the red color is included in a specific pixel by simple comparison of values in the red component, but it is not easy to recognize that the final color of the pixel is still red after combining it with blue and green components. Thus, we take advantage of HSV color space, and convert our RGB images into HSV color space. The user may define minimum and maximum thresholds for ranges of “hue”, “saturation” and “value” to pick specific pixels, and create a mask image for them. Based on its cylindrical geometry, we may choose minimum and maximum pure colors for Hue to find any combination of them (e.g. finding orange color between red and yellow). Similarly, Value and Saturation assist us to accept or reject that color based on the amount of colorfulness or brightness. This step can be repeated a few times if more regions for separate range of colors must be removed or kept untouched (e.g. blue or green color of the flames in specific frames). Our mask can be created using the following formula:

$$M(x,y) = \begin{cases} 1 & \begin{cases} hue_{\min} \le p(x,y) \le hue_{\max} \\ saturation_{\min} \le p(x,y) \le saturation_{\max} \\ value_{\min} \le p(x,y) \le value_{\max} \end{cases} \\ 0 & otherwise \end{cases} \tag{1}$$
$$hue \in [0,360], \quad saturation \in [0,1], \quad value \in [0,1]$$

where $p(x,y)$ is a pixel on the image in HSV space at pixel location of (x,y), and $M(x,y)$ is our binary map that can be used to extract our contours.

Morphological operations can be employed to filter out potential noises. Our application allows the users to apply well-known filters and operations (e.g. Dilation, Erosion, Opening, Closing, Gaussian smoothing, contour optimization, hole removal, etc.) based on a bottom-up layer style for minor improvements, to obtain a consistent contour for the entire video. It should be considered that all the employed techniques will be applied to the entire frames of the video to eliminate or reduce the need of manual changes for specific frames. As a result, more inspection of techniques was required in comparison to regular image based approaches.

Using Canny edge detection algorithm, binary thresholding and border following algorithm [19], contours were retrieved from the binary mask we created. Fire is not always a closed shape, and it can split or create a hole in a few frames. We therefore store our contours hierarchically in a tree, to be able to remove unwanted contours based on a depth level system assigned to each of them. The area inside each contour is also calculated and stored. We call this as "*Hole Removal*", which can be area-based or depth-based, to eliminate specific small contours that appear unexpectedly during an entire video. Finally, Douglas-Peucker algorithm [6] is used to optimize and approximate the contours whenever required. Fig. 3 shows the process of our contour detection. A depth level of "0" means the contour is separate from other detected ones, and depth level of "1" indicates a contour which is one level inside another one, and so forth.

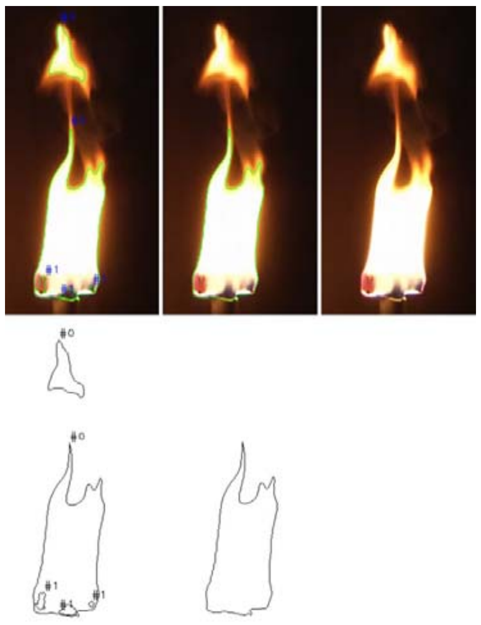

Fig. 3 An example of contour detection. The top row: (Left) all detected contours with any possible depth levels, (middle) contours with depth levels of only zero, in addition to area-based restriction of specific number of pixels, (right) the original image of fire. The lower images illustrate the corresponding contours of the top row separately.

3.3 Hermite Background Removal

As we already discussed, luminosity of flames brightened both background and smoke. We are planning to diminish that undesirable brightness by a darkening process outside the contours. It is important to preserve the smooth shape of the fire, specifically around contours. We are trying to take advantage of this smoothness to produce images with soft corners while calculating eigenfires. In majority of previous works that PCA was used such as [20][22][23], the images had been modified in such a way that the background was ignored outside the target with sharp boundaries. The reason is that they are mostly interested in detection and comparison of features inside the objects. The difference is obvious from comparison of the average of training images. In our initial experiments, we removed areas outside the contours by directly assigning zero values to every pixel, and instead we could only produce flames with numerous layers of sharp edges, even by using a huge training set of images. It should be noted that eigenfires are highly relevant to our average image, and therefore we can avoid unexpected artifacts during reconstruction while using small number of dimensions.

After experimenting with linear, cubic and Hermite curve approaches, we realized that the second Hermite basis function yields smooth transitions from pixels with very high intensities to the pixels with low intensities. The equation of this basis function and our intensity criteria are as follows:

$$\mathit{Intensity}(S) = -2S^3 + 3S^2 \qquad S \in [0,1] \tag{2}$$

$$S = \frac{p(x,y) - I_{\min}}{(I_{\max} - I_{\min})} \qquad \mathit{if}\ I_{\min} \le p(x,y) \le I_{\max} \tag{3}$$

where S is a number in range of $[0,1]$ which is based on two values of maximum and minimum intensities, and are set by animators. The value of any pixels with intensity of higher than $I_{\max}$ does not change, but any pixels lower than $I_{\min}$ become zero or black. Any pixel between them will be gradually darkened in respect to $\mathit{Intensity}(S)$, and finally produces a smooth transition of intensities.

To get better results, we can consider our calculated contour of the flame as a bitmap mask and then ignore any changes of intensities inside the contour region. The result of this technique is shown in Fig. 4.

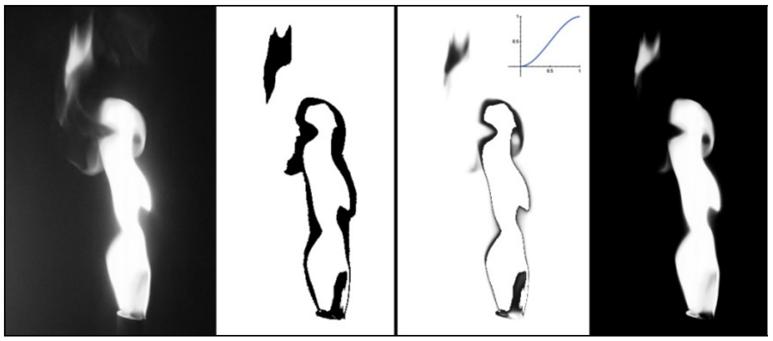
(a) (b) (c) (d)

Fig. 4 Hermite Background removal process. (a) Original RGB image converted to gray to be prepared for PCA, (b) a mask by choosing $I_{\max} = 220$ and $I_{\min} = 110$, (c) modified pixels obtained from multiplying *Intensity(s)* with current pixels of our gray image located in black regions of our mask, (d) final image with smooth borders

3.4 Direction Normalization

Although our cameras' positions are fixed and the root of a flame does not move during an entire video, we are combining multiple videos with different dimensions and volume. Their fuel is also replaced with a new material, and therefore it is required to be repositioned and aligned at the center. Scaling and translation are performed manually once for each video which is not tedious with our user-interface. Nonetheless, translation could be performed automatically by tracking static objects in the view (e.g. burning objects or upper part of the surface beneath the fuel) and thresholding.

Using the vertices of the calculated contours, we are able to eliminate direction factor from fire, which is performed by fitting a line using weighted least-squares algorithm. After finding this line, we should trim it to calculate its length. Because it is not clear which point is the first or the last point along this fitted line, all the points should be projected into the line, and therefore farthest points are the end points for cutting the line. Using this line, we can rotate our image around the start point that is located in lower part of the line. Douglas-Peucker algorithm [6] is employed to simplify the contour, and to eliminate unnecessary vertices. This is illustrated in Fig. 5.

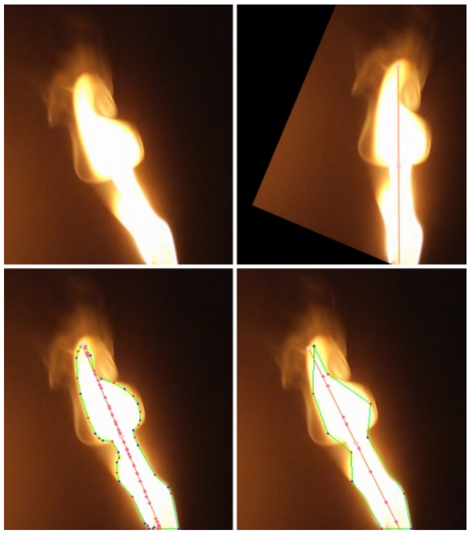

Fig. 5 Pose normalization. Top row: (left) original image, (right) auto-rotated image after pose normalization by fitting a line. Note: the entire background will be darkened based on section 3.3. Bottom row: two contours with different levels of optimization that show almost the same direction. The fitted line is illustrated with a pink color, including projected data points on it

The direction of flame should always start from somewhere around fuel or burning object, but the line might be constructed incorrectly in another direction if the width of a flame span horizontally and become shorter in height. We suggest three techniques as remedy to this problem:

1) The best approach is optimizing the contour and converting it to a low poly shape, in such a way that it only contains two points at the root; therefore it will perfectly create the desired line for the entire images with a very low error rate. 2) A user may set a region of interest before line fitting procedure, and increase the number of the points in that area for the contour. 3) Special weights may be assigned to data points in the same region of interest.

Although auto-rotation yields interesting results for shape recognition, we decided to disable it for video samples chosen for this paper after running a few experiments. The reason is that the direction factor became part of our main eigenfires as an important feature. We explain it further in next sections.

4 Eigenfires and Their Characteristics

Principal components, represented as eigenfires, are calculated for 2700 frames of our three fire samples based on algorithms described in [20]. The images are cropped to 410x670 resolutions to speed up the process by avoiding unnecessary calculations for the static background. Considering N as dimension of eigenvectors, each image is dealt with as a single data point in $N = 274{,}700$ dimensional space. Thus, the principal components are a series of vectors of N dimensions, and because the number of images in space is less than the dimension of space, the number of eigenfires (E) will be equal to the number of our data points (fire images) minus one.

Among the first 30 eigenfires which clearly express significant features of our sample flames, the first 10 are mainly used as the foundation to reconstruct diverse shapes of flames with smooth edges. By projecting fire images into its eigenfire, a vector of weights will be constructed. Each weight represents contribution of its eigenfire to form a fire. Since assigning a negative or positive value for weights creates diverse styles of fire, we take advantage of that to scrutinize eigenfires, and later to reconstruct a new flame which can be a combination of three flames. The experiments suggest that with $E = 700$ eigenfires, we can properly identify similar frames through recognition stage, since our reconstructed images only lose some pixels (noise appearance) inside the boundaries of flames which is negligible in detections. Likewise, employing a higher numbers of eigenfires but still below 50% of the total number of images (e.g. $700 < E < 1{,}200$) would be sufficient for high quality reconstructions or to obtain ridge effect. The reason is because losing some data here means appearance of small bumps which is acceptable for fire modeling.

Fig. 6 displays the average of fire images. The first nine eigenfires and another three with lower eigenvalues are showed in Fig. 7. For better understanding of how negative or positive values of weights may contribute to each fire image, we give a simple but inaccurate explanation. First, 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, which is the corresponding eigenfire, using dark pixels of that eigenfire visible in Fig. 7. Similarly, a positive value fills image *I* using bright pixels of that eigenfire (white pixels). The gray regions remain almost unchanged, such as background. Although this is not a precise definition, it is what you may visually perceive by changing weight for one specific eigenfire. The accurate definition of a weight (w) and projection is subtracting an image *I* with average of images, and then multiplying it with its eigenfire:

$$w = eigenfire^{T} \; (I - I_{Average}) \tag{4}$$

The weights are numbers in range of [-15000, +15000] for our samples, and this range changes based on the training set of images.

In our system, the eigenvectors are stored in separate 32-bit floating point images, and then converted to 8-bit gray images to be displayed as eigenfires for visualization purposes. Since our eigenvectors may contain very small decimal numbers, we need to perform the following scaling and shifting on the pixel values for conversion to 8-bit images:

$$scaling = \frac{255}{pixel_{\max} - pixel_{\min}} \qquad shifting = \frac{-255\,pixel_{\min}}{pixel_{\max} - pixel_{\min}} \tag{5}$$

where $pixel_{\min}$ and $pixel_{\max}$ are respectively the smallest and the biggest values (slightly larger than zero) among *N* dimensions in each eigenvector.

Short descriptions of a few effective features are provided in Table 1 for their relative principal components. Vague contributions are left blank in this table. We reconstructed new flames with customized weights to observe the influence of each eigenfire for this table, which is also partially predictable from Fig. 7.

To better understand the pattern between data points in PCA subspace, weights of the first 10 eigenfires are illustrated in Fig. 8 using charts. For each video sample, 31 consecutive frames are depicted in clustered columns for comparison of the patterns and changes of the weighs.

The data points of the first three principal components are also visualized in 3D, in Fig. 9. This figure shows that the data points related to each video are distributed in separate locations far from each other. The center of the coordinates system is a vector of weights with zero values that produces an average image if we reconstruct a point at that location. We labeled the data points corresponding to each video with the same labels as already mentioned in Fig. 2.

Table 1 Features of the first ten principal components

Principal Component	Feature Description	Positive weight	Negative weight
1	Direction of fire (a_1)	Toward left side	Slightly toward right side
2	Direction of fire (a_2)	Toward right side	Slightly toward left side
3	Height and width	Increasing height, decreasing width	Decreasing height, increasing width
4	Opening the flame at the tip (b_1)	Changing the tip to two wide branches	-
5	Shape of the tip of flame	-	Cone shape
6	Volume	Larger volume	-
7	Opening the flame at the tip (b_2)	Two sharp branches	Three sharp branches
8	Hole (c_1)	A hole at the tip (two branches)	Two holes in fire
9	General shape deformation	-	-
10	Hole (c_2)	A hole in middle	-

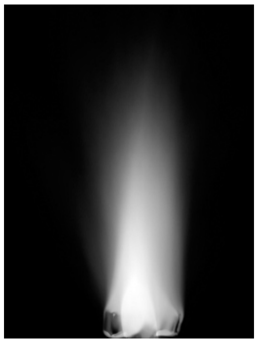

Fig. 6 The average of fire images. A vector of weights with all values equal to zero will result this average image as well. Choosing a positive or negative value for weights deforms this shape, based on the corresponding eigenfire.

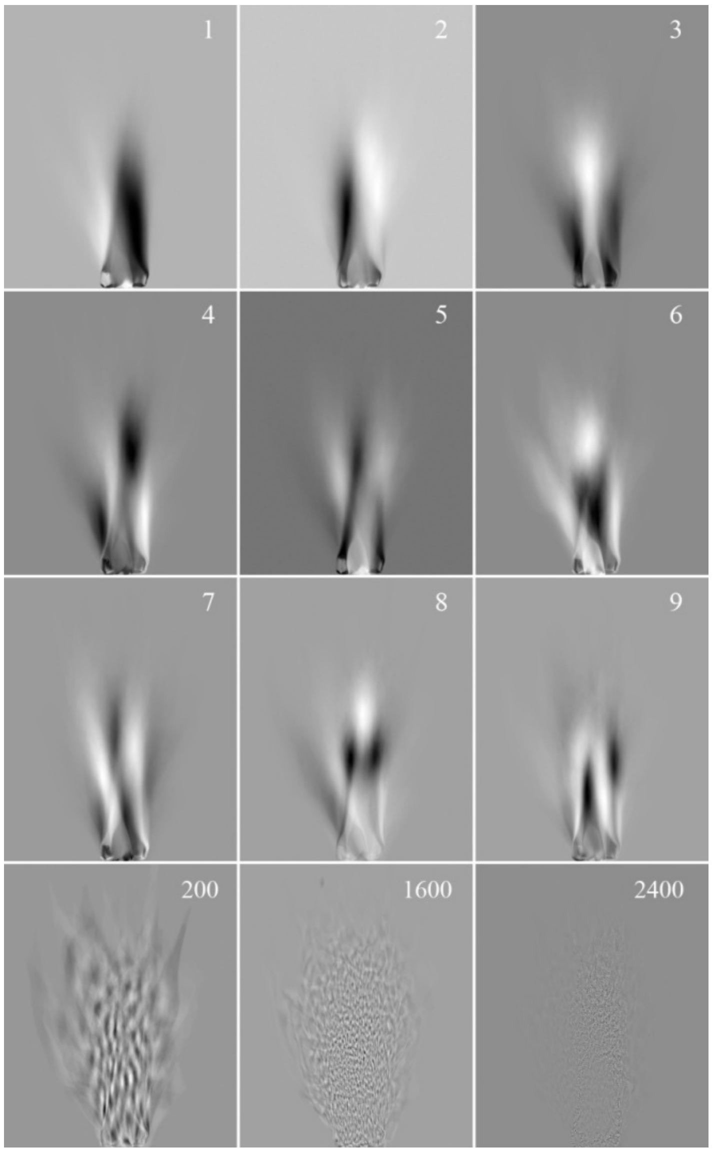

Fig. 7 Top three rows are showing the first nine principal components viewed as eigenfires. The bottom row from left to right: principal components of 200, 1600 and 2400. It is obvious from this row that eigenfires with smaller eigenvalues produce fine details and edges of our flames, depicted as some sort of noise

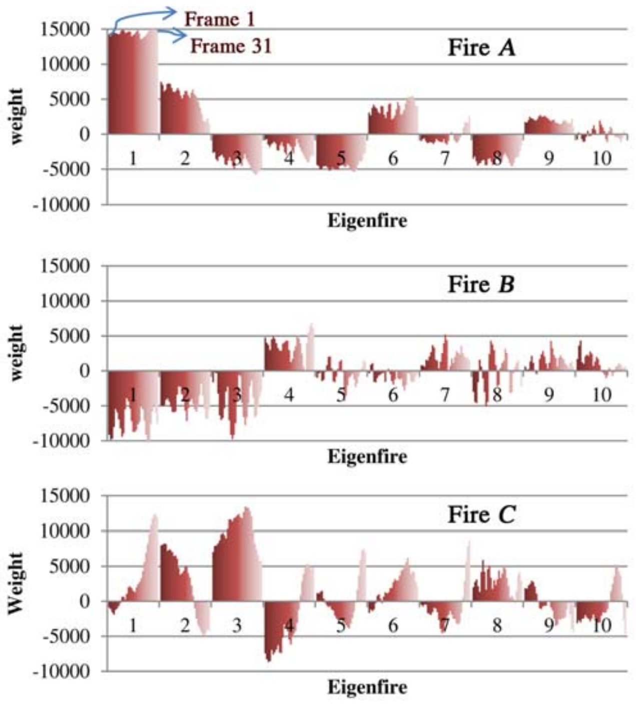

Fig. 8 Weights assigned to 31 consecutive frames of our three sample videos are shown in clustered columns using the first 10 eigenfires. The darkest color on the left side of each clustered column starts with frame number 1, and brightest one ends with frame number 31 on the right side, which are also indicated on the figure

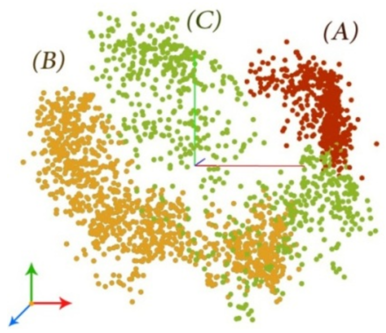

Fig. 9 All *M* data points projected in PCA subspace, illustrated as a series of weights for the first three eigenfires. Red, green and blue axes are the first, second and third principal components respectively. Red, yellow and green points are labeled with their relative fire's name (remember: Fire *A*, *B and C*)

5 Eigen Fire Weight Adjustment for New Fire Creation

A few reconstructed images, which are projected into multiple dimensional spaces, are depicted in Fig. 10. The Hermite background removal method is utilized to diminish undesirable artifacts from reconstructed images with lower dimensions, and the quality is enhanced considerably.

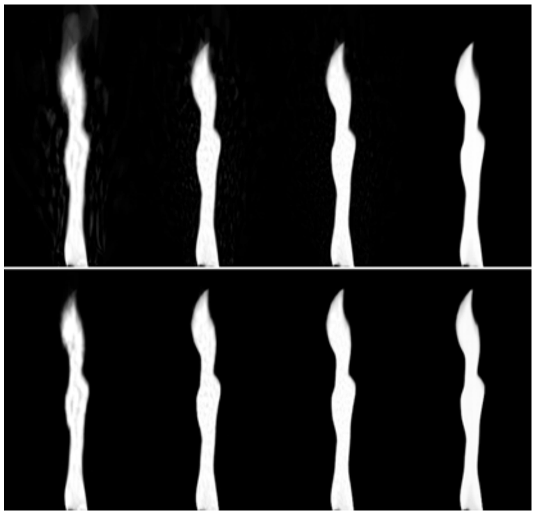

Fig. 10 Projecting fire images into different dimensional spaces. Images of the first row from left to right are reconstructed using: 200, 700, 1100 and 2699 eigenfires. The Second row shows that the quality of the same images are improved with our Hermite background removal

In this approach, animators may load the calculated weights of a specific image, and then manually increase or decrease the ratio of certain eigenfires via user-interface to deform the shape of a flame. This way, the new image is displayed in real-time for validation. Since our first 50 eigenfires carry major features of flames, it is not even very tricky to start from scratch, by assigning zero values to all the weights at the beginning, and then allocating new values for various eigenfires. The user may take a look at the generated eigenfires of the database, to find out the influence of assigning a positive or negative value to each eigenfire. Fig. 11 illustrates a manually constructed image with manipulation of only 7 eigenfires (0.002% of dimensions!).

Assigning small values to any eigenfire larger than 400 (e.g. $400 < E < 2700$) produces some fine details, which is similar to applying procedural noises. An interesting property here is that by assigning small values to weights of those ranges of eigenfires, most changes will appear over the white regions. It means our black background usually remains unaffected or less affected. The drawback of manual weight adjustment is difficulty of creating realistic animations.

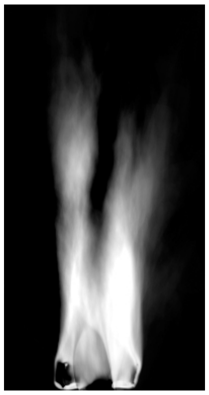

Fig. 11 Manual weights are assigned to eigenfires of 1, 3, 4 and 10 for shape, and eigenfires of 50, 100 and 650 for distortions and some details. The rest are set to zero values

6 System Realization

We ran our experiments on a system with Intel Core i7-2600k 3.4GHz CPU and 8GB 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 visualization. The 64-bit implementation allows the users to process very large video files, and virtual memory can take care of required memory which might not be available physically on a working system.

Fig. 12 shows a screen-shot from our application. Using the optimized number of E=700 eigenfires (74% compression), in addition to darkening background and other functionalities of our system, it takes around 160 ms to reconstruct a new image, which is at least 3 times faster than using all dimensions (450 ms using all 2699 eigenfires). It takes around 2 hours to prepare the initial database of eigen-fires for 2700 images of 410x670. This total time includes both PCA calculations and projecting all training images into PCA subspace, which are only performed once and then saved as a library of fire. For 1000 images of the same resolution, the total PCA calculation time is only 12 minutes.

7 Conclusions and Future Work

We introduced a novel approach to procedurally model fire by extracting the pat-terns of the real flames using eigenfires, which is not based on unrealistic random noises. We also proposed a few techniques, such as Hermite background removal and contour detection, to improve the quality of the final images. As a result, we can compress the database to 26% of entire size with dimensional reduction. Our methods are fast and changes can be followed in real-time. Our research proves the fact that PCA can be extended to further topics with similar concept of reconstructing new images out of an extensive database.

Comparison of our sample videos recorded at both 30 fps and 60 fps reveal some subtle but important changes of model in majority of frames. Accordingly, more accurate results can be obtained from high-speed cameras that provide 1920x1080 full HD resolutions at frame rates of at least 300 fps, such as Phantom HD Gold, Photron FastCam SA2 or recently Sony NEX-FS700 camera. Although these cameras can record up to 2,000 fps at HD resolution, we do not suggest capturing higher than 300 fps, as it will dramatically increase the processing time of learning stage, and there will be very little variation among the shapes of consecutive frames in fire, which can be ignored.

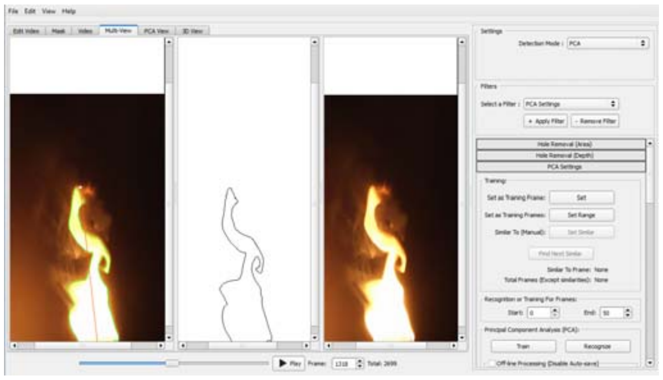

Fig. 12 A screen-shot from our system, showing recorded videos and contour detection

References

[1] Adabala, N. & Hughes, C. (2004), A Parametric model for real-time flickering fire, in 'Proceedings of Computer Animation and Social Agents(CASA)'.

[2] Amarasinghe, D. & Parberry, I. (2011), Towards fast, believable real-time rendering of burning objects in video games, in 'Proceedings of the 6th International Conference on Foundations of Digital Games', ACM, New York, NY, USA, pp. 256--258.

[3] Beaudoin, P.; Paquet, S. & Poulin, P. (2001), Realistic and controllable fire simulation, in 'No description on Graphics interface 2001', Canadian Information Processing Society, Toronto, Ont., Canada, Canada, pp. 159--166.

[4] Crane, K.; Llamas, I. & Tariq, S.Nguyen, H., ed., (2007), Real Time Simulation and Rendering of 3D Fluids, Addison-Wesley, chapter 30.

[5] Dardas, N.H.; Petriu, E.M.; , "Hand gesture detection and recognition using principal component analysis," Computational Intelligence for Measurement Systems and Applications (CIMSA), 2011 IEEE International Conference on , vol., no., pp.1-6, 19-21 Sept. 2011

[6] Douglas, D. H. & Peucker, T. K. (2011), Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature, John Wiley & Sons, Ltd, pp. 15--28.

[7] Fuller, A. R.; Krishnan, H.; Mahrous, K.; Hamann, B. & Joy, K. I. (2007), Real-time procedural volumetric fire, in 'Proceedings of the 2007 symposium on Interactive 3D graphics and games', ACM, New York, NY, USA, pp. 175--180.

[8] Harris, M. (2004), Fast Fluid Dynamics Simulation on the GPU, in Randima Fernando, ed., 'GPU Gems', Addison-Wesley, , pp. 637--665.

[9] Hong, J.-M.; Shinar, T. & Fedkiw, R. (2007), Wrinkled flames and cellular patterns, in 'ACM SIGGRAPH 2007 papers', ACM, New York, NY, USA.

[10] Horvath, C. & Geiger, W. (2009), Directable, high-resolution simulation of fire on the GPU, in 'ACM SIGGRAPH 2009 papers', ACM, New York, NY, USA, pp. 41:1--41:8.

[11] Lamorlette, A. & Foster, N. (2002), Structural modeling of flames for a production environment, in 'Proceedings of the 29th annual conference on Computer graphics and interactive techniques', ACM, New York, NY, USA, pp. 729--735.

[12] Melek, Z. & Keyser, J. (2002), Interactive Simulation of Fire, in 'Proceedings of the 10th Pacific Conference on Computer Graphics and Applications', IEEE Computer Society, Washington, DC, USA, pp. 431 - 432.

[13] Min, K. & Metaxas, D. (2007), 'A combustion-based technique for fire animation and visualization', The Visual Computer 23, 679-687.

[14] Nguyen, D. Q.; Fedkiw, R. & Jensen, H. W. (2002), Physically based modeling and animation of fire, in 'Proceedings of the 29th annual conference on Computer graphics and interactive techniques', ACM, New York, NY, USA, pp. 721--728.

[15] Pegoraro, V. & Parker, S. G. (2006), 'Physically-Based Realistic Fire Rendering"In Eurographics Workshop on Natural Phenomena, E. Galin and N. Chiba (editors)', 237--244.

[16] Stam, J. (2003), Real-Time Fluid Dynamics for Games, in 'Proceedings of the Game Developer Conference'.

[17] Stam, J. (1999), Stable fluids, in 'Proceedings of the 26th annual conference on Computer graphics and interactive techniques', ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 121--128.

[18] Stam, J. & Fiume, E. (1993), Turbulent wind fields for gaseous phenomena, in 'Proceedings of the 20th annual conference on Computer graphics and interactive techniques', ACM, New York, NY, USA, pp. 369--376.

[19] Suzuki, S. & Abe, K. (1985), 'Topological structural analysis of digitized binary images by border following', Computer Vision, Graphics, and Image Processing 30(1), 32-46.

[20] Turk, M. & Pentland, A. (1991), 'Eigenfaces for recognition', J. Cognitive Neuroscience 3(1), 71--86.

[21] Vanzine, Y. & Vrajitoru, D. (2008), Pseudorandom Noise for Real-Time Volumetric Rendering of Fire in a Production System, in 'Volume Graphics', pp. 129-136.

[22] Xi, P.; Lee, W.-S.; Frederico, G.; Joslin, C. & Zhou, L. (2006), Comprehending and transferring facial expressions based on statistical shape and texture models, in 'Proceedings of the 24th Computer Graphics International ', Springer-Verlag, Berlin, Heidelberg, pp. 265--276.

[23] Xi, P.; Lee, W.-S. & Shu, C. (2007), A Data-driven Approach to Human-body Cloning Using a Segmented Body Database, in 'Proceedings of the 15th Pacific Conference on Computer Graphics and Applications', IEEE Computer Society, Washington, DC, USA, pp. 139--147.