Three Dimensional Fire Simulation Based on Visual Learning of Image Features

Chungnan Lee, Weilun Tai, Dajing Zhang-Jian, and Wenchieh Hsieh
Department of Computer Science and Engineering, National Sun Yat-Sen University
E-mail:cnlee@mail.nsysu.edu.tw, weilun424@gmail.com, salmoner.tw@yahoo.com.tw, and t49102@gmail.com

***Abstract* — The natural phenomena simulation in computer graphics is commonly achieved by the procedural methods or the physics model. However, these approaches are hard to directly approach the visual experience. On the other hand, the image reconstruction works can provide the outcome based on the real images but lack of interactivity and efficiency by using the image resource. In this paper we propose a novel method that enhances the fire simulation effect using the visual learning of image features and generates continuous animations by integrating with procedural methods. We first obtain the dynamics of fire contour by binarization and edge detection. The information extracted from images is gathered into a set of feature data called fire profile. To generate a long sequence of fire animation from a short clip of fire video, we propose two approaches of visual learning to utilize fire profile to produce continuous animation. One is to use the fire image to setup a color value lookup table which contains the average color value of the fire spatial divisions; the other is to design a state machine for describing fire wiggling movement that can generate effects based on user's input. The proposed method can raise not only the visual reality but also the interactive ability compared with the existing work.**

I. INTRODUCTION

In the past few decades, developing a visually convincing model of fire, smoke, and other gaseous phenomena is always a challenging issue in computer graphics. Especially people have developed a variety of simulations to model fire. However, some of them have heuristic characteristics due to the parameters that are calculated and manipulated through some human defined procedural methods. For this reason, we believe that the contents of the video which contains the fire image can enhance the visual feelings of simulation. Meanwhile, some works [1][2] to reconstruct fire from image are only focus on recovering the appearance of fire by the governed images and impossible to use the production in some applications that need the demand of interactivity.

Therefore, we proposed a novel method that has two properties: One is to utilize visual information from video clips to generate fire animation, and the other is to combine visual effect with a particle system to continually generate fire simulation to overcome the short clip of video sequence.

In the paper, the proposed method including image processing is to extract major features from images and reconstruct a three dimensional effect based on the extracted features. Hence, users may get the effect without taking a complicated mathematical model or additional modeling method. The proposed method is implemented into a semi-automatic animation program. The system scenario is depicted in Fig. 1. The video can be transformed into image sequence files in advance, and stored into the system for preprocessing. The intermediate file of the system contains the feature data that are collected from the processing of the image sequence and further sent to the 3D rendering stage for visualization.

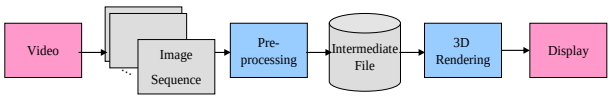


Fig. 1 System scenario

While modeling fire phenomena in computer graphics, currently there is no overwhelming method among the existing approaches. Animators can choose the model according to their objective, such as accuracy or efficiency. For the online application, the procedural methods are commonly used in industry. Yet it is regularly called in question about the heuristic property, which means the procedural method is developed without the explicit visual definition. On the other hand, the fluid mechanics methods are precise in the numerical computation and good at presenting fine effect. However it is not suitable for real-time application due to its huge computing load. And these two types of simulations are totally relied on the simulating model so that ensuring the final production realistic is impossible.

In the proposed method, it first gathers necessary information from image processing techniques. Then the information will be collected into a set of data about the features of fire. For using limited images effectively, we developed a learning mechanism to make the short-time

image sequences to be used in long-time. The result generated by the learning mechanism is combined with procedural methods to render the fire animation. Then the production come out through the feature statistic learning module and the modified particle system is more realistic than the traditional fire simulation methods and have the ability of interaction relative to the reconstruction-style methods.

The rest of this paper is organized as follows. In Section II, we briefly describe the background materials and related work. Section III describes the proposed method for the fire scene simulation. Implementation and rendering results are given in Section IV. Finally, conclusion is given in Section V.

II. RELATED WORK

This section reviews certain modeling and rendering methods for simulating fire phenomena in computer graphics. The key impact on simulation result is the accuracy of modeling and resolution of whole portrait. Before starting to investigate the related work, one needs to understand that the work for fire phenomena can be categorized into modeling and rendering that have different ideas in improvement and development of algorithms.

A. Modeling Methods

Modeling realistic fire animation needs to define the motion of flame adequately. In previous approaches, the modeling methods for describing chaotic movement of flame are to combine the procedural components, solve fluid mechanics equations, and represent the integral statistics. However, those works are for modeling other natural phenomena, such as fog, smoke, and fluid will not be discussed here. This section mainly reviews the fire-related simulation approaches.

Procedural models for the simulation of fire are based on heuristics rules that govern the dynamics of simple primitives. These methods in [3][4][5] are not strictly based on physical principles, though tuned appropriately they can nonetheless generate interesting visual effects. Furthermore, procedural models often have the advantage of being computationally efficient.

The main drawback of these methods is that controlling the phenomena can be difficult, with many brittle parameters to tweak in order to achieve reasonable results. Moreover, extending the models to obtain new effects or to interact with other elements in the scene is often impossible.

The earliest computer graphics fire model was presented by Reeves [4]. The model uses a large number of particles to animate a fire engulfing a planet. Particles can easily represent fuzzy objects and the adjustment to attributes of particles can receive varieties of outcome. Chiba et al. developed a 2D model of fire based on a procedurally defined turbulence field [3].

Takai et al. proposed another method for fire simulation by cellular automata [6], which is occupied by one fire particle and labeled with mass velocity and discrete velocity. The cellular automata can produce extremely complex structures from the evolution of rather simple local rules. Dobashi et al. [7] used a special cellular automata model to implement a realistic animation of clouds. For each particle, the movement is controlled by randomized transition rule. Wei et al. [8] adopted LBM (Lattice Boltzmann Model) to substitute cellular automata as the fire dynamic model. LBM was proposed by John Von Neumann as formal models of self-reproducing organisms. Several variables are defined at each cell to indicate whether there are microscopic packets moving in a certain direction. Balci et al. [9] modeled the kinematics of these phenomena by a mass-spring system, and used texture sequencing with variable speeds and transparencies that depend on the speed of the vaporized fuel to present turbulant visual dynamics.

For generating realistically visual effect of fire, the models adopted by most animators are based fluid mechanics rules. The developers can create visually plausible result without considering the implementing details. However, this method has some constraints on setting boundary conditions and keeping system stable. Furthermore, physics based model is also hard to achieve direct control for multiple purposes.

In particular, this method requires huge computation that is at least $O(n^3)$ with the resolution of the simulation. So far this innate problem only can be solved by hardware acceleration. In addition, the solver must take care of instability in a relatively sophisticated manner. To execute in real-time, models of fire typically retain the incompressibility assumption but account for the expansion of hot gases as the result of external forces [10]. Hong et al. [11] modeled fire by combining the Navier-Stokes equations with the level set method and jump conditions for simulating the reaction front. Asymptotic theory shows that one can obtain more entertaining vibration and fully hyperbolic equations for the evolvement of level set. Nevertheless, they also showed how to utilize the DSD (detonation shock dynamics) framework in the simulations of flames and fire to receive interesting visual experience such as wrinkles and cellular patterns.

B. Rendering Methods

Rendering fire demands some kind of volume rendering because of its visual characteristics such as opacity and transparency. However, we examine general volume rendering techniques via graphics hardware for rendering fire, and more complicated models that treat the fire as a source of illumination and capture mirage-like refraction effects. These rendering techniques [7][12] are described in the context of more general techniques for rendering smoke and other gases, even though certain effects like scattering are ignored by nearly all image formation models of fire.

Scientific visualization methods [13][14] are mainly used to deliver information effectively, rather than to describe the phenomena that statistic in a rational way. The most important conditions for rendering fire by scientific visualization are to keep essential information for the domain, extract the information, and utilize acceptable rendering skills.

Direct volume rendering is frequently adopted as the approach for rendering fire. In some particularly cases that the volume of fire is represented as cylindrical rings [15] the integration along streams of light can be accelerated. If the primitives of density field are defined by some type of implicit expression such as Gaussian blobs, constructing lookup tables can help deliver additional speed [12].

There are many sophisticated and quicker volume rendering methods have been developed in earlier works. One is to speed up ray tracing, the simple way is to end up before it decreases gradually to zero. Another is splatting [16], that prevents costly expense on 3D interpolation for volume rendering. Stam suggested another interesting approach to render turbulent volume fields [17]. Under his stochastic rendering framework, the scene to be rendered is represented as a random field with known statistics. These statistics are then shifted in a systematic fashion from the modeling phase to the rendering phase. This allows relatively simple primitives to be used in modeling, but more complex images to be obtained from the rendering.

Certain techniques that exploit fast graphics hardware enable the rendering of fire at interactive rates. Perhaps the simplest technique for rendering fire quickly involves applying movie loops as textures to billboards [18]. The set of textures can be generated using spectral turbulence methods, or by extracting pieces from real images of fire. King et al. combined this texture-based approach with traditional volume rendering using splatting [19]. They used a coarse voxel grid, where each voxel was assigned an index into the set of textures. By manipulating the texture indices as if they were particles, they gave the impression of smooth motion. When performing splatting, the kernels were modulated with the appropriate texture corresponding to that voxel. In contrast to the billboard method, their method is effective even when the viewpoint is inside or very close to the fire. For added realism, one would like the fires we are modelling to act as light sources and cast illumination into the scene. Takahashi et al. proposed splatting method that uses similar idea to modify volume rendering [20].

C. *Motivation*

Based on the observation of the existing related work, we propose a method to overcome some shortcomings listed as follows:

- Visual reality of fire simulation: Lacking of visual reference for convincing the viewers and cannot expect the correctness of result before the development.
- Usage of fire simulation: Commonly concentrate on creating an accurate model or representation, and it is neglected to provide interactivity between users and the virtual environment.

Therefore, this paper utilizes the content of real world images and makes use of their features to enhance the reality of rendering. The major contributions of the proposed method are as follows:

- To extract fire features from a real world video for enhancing reality of fire simulation.
- To produce continuous result without a great deal of resources but also can be applied to real-time application easily.
- To generate fire effect based on images instead of only adjusting the parameters of a particle system by artists, it can reduce the time on trial and error to speed up the fire simulation.

III. PROPOSED METHOD

In this section, we propose a method that first extracts the image features using the image processing techniques from the real world image instead of computing from physics equations or mathematical models. Then we design a statistical learner to fully utilize the flame features in rendering efficiently. The learner governs the life of particles that is generated from a particle system. The system overview is illustrated in Fig. 2. It can be divided into two main stages. The first stage is preprocessing that focuses on acquiring the properties (such as shape and color) and gathering them into a fire profile. Because the processing time is too long for computing in real time, the preprocessing stage will be executed in offline. Once the fire profile is obtained, one can render animation online. During the execution of rendering stage, the feature statistics learner inside the rendering system governs how the particle system to use fire profile. The feature statistics learner will exploit the fire profile to build up a color lookup table and provide a state machine for changing the leaning state of fire. Then a user can input the external force to change the effect of wiggling movement.

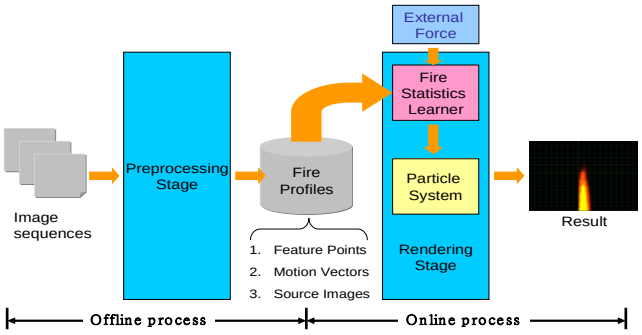


Fig. 2 System overview

A. *Preprocessing Stage*

The flow diagram of preprocessing is given in Fig. 3. In this stage, the features including the burning fountainhead, color, and boundary are extracted. Then these features are gathered into a fire profile for the upcoming rendering step. The advantages of this approach not only can find promising fire trends for designers, but also can turn natural simulation into a new type rendering technique. The initial process of this stage applies a thresholding and Sobel edge detection. After the processing steps, the image will be transformed into a contour image of fire. Features and motions are further extracted for fetching the fire information in the image. All the steps mentioned above are the essential for acquiring the fire profile features. The fire profile is used in the rendering stage as the government of modeling the shape and luminance of flame.

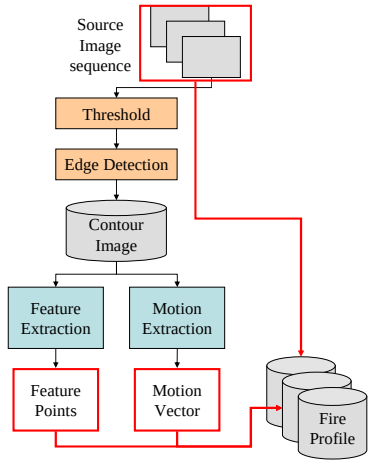
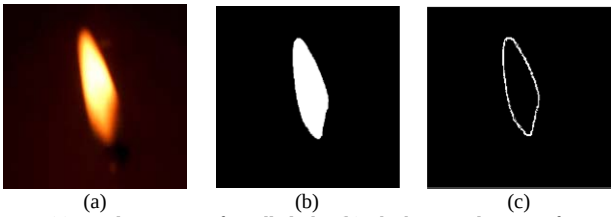


Fig. 3 Processing flow of preprocessing stage

The bilevel thresholding is used to segment fire region from images. Fig. 4(a) is the color image which captured from general digital camera, and the corresponding outcome of binarization process is shown in Fig. 4(b). After thresholding, the Sobel edge detection is used to calculate the gradient of the intensity of every texel in an image. Fig. 4(c) is the result after the Sobel operation.



(a) (b) (c)

Fig. 4 (a) A color picture of candle light, (b) The binarized image of source image, and (c) The image that applied the Sobel operation to Fig. 4(b)

Several researchers have studied this issue based on porous bed gas burners [21][22]. McCaffrey showed that the ”fire plume” above a 30 cm square burner consisted of three distinct regimes [22], precisely: (1)The near field, above the burner surface, where there is persistent flame and an accelerating flow of burning gases (the flame zone); (2)The intermittent zone, a region in which there is intermittent flaming and a near-constant flow velocity; (3)The buoyant plume is characterized by decreasing velocity and temperature with height.

The proposed algorithm uses these three regimes principle as in [22]. When the combustion reaction proceeds without involving any other factor, the movement of flame is upward direction that is created by the buoyant force. Even if there are some external forces existed in this burning system, the persistent flame regime will remain stable. In the feature point extraction, one needs to find six extreme points of fire in the current frame. Among these six points, five points are used for building up a curve-like emitting region, and the rest one is used for subscribing the fire peak point. Let $I(x,y)$ represent a pixel at spatial location(x,y) on I, where I is the pixel set of fire contour which is formed by the processing steps in previous subsection. *Left(I)* and *Right(I)* as shown in Fig. 5 are found at the contour which is under the threshold of persistent regime, here we set it as one fifth of the flame height. The proposed method is to seek the maximum width of fire in the persistent regime, and take the two ending points of the maximum width as *Left(I)* and *Right(I)*. To find *Top(I)* and *Bottom(I)* one can assign the highest point of flame to the extremely top point, and the lowest point to the extremely bottom point. *Top(I)* is used for assigning the fire peak point that it is the ending point of curving path of particles. However, the bottom area of flame may be formed with multiple inflection point. Then one can add two middle points *MidLB(I)* and *MidRB(I)* between *Bottom(I)* with *Left(I)*/ *Right(I)* in order to fit any possible shape. Depending on these six points one can formulate the fire shape and the dynamic path of all particles.

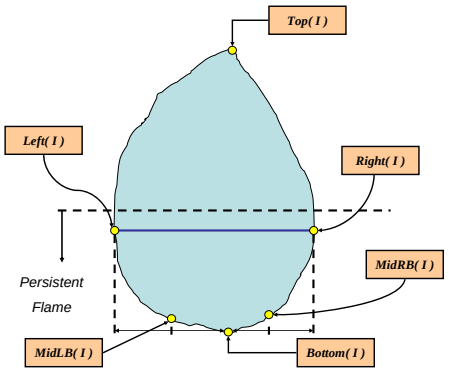


Fig. 5 Six extreme points of fire per frame

B. *Feature Statistics Learner*

If one wants to generate animation continually, this property may be an obstacle because of the limited time of

source image. Therefore, to generate a sequence of fire animation based on fire profile and a particle system (a procedural process) can eliminate this drawback.

Generally speaking, the most two important factors about influencing the human visual experience is the color and the dynamics of an object, we will handle the two issues independently in our implementation. However, only the features come from images are available, it is essential to learn from the fire profile for simulating fire. We develop a learning mechanism consisting of a fire status state machine and a color lookup table based on the fire profile to approach the visual experience of fire as shown in Fig. 6. During of rendering stage, the feature statistics learner will be executed before go into the particle system. In the following subsection we will discuss how the learner works and what the outcome is generated.

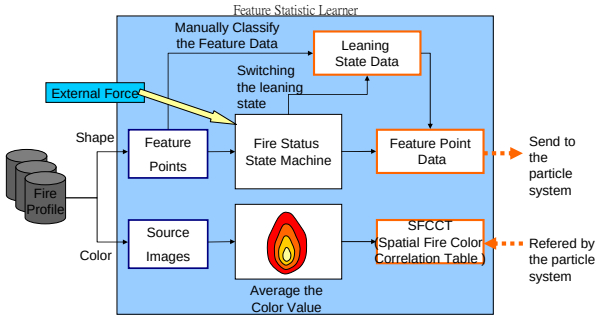


Fig. 6 Mechanism of feature statistic learner.

A number of feature point data collecting from the fire image are gathered into the fire profile. Depending on the observation of the fire dynamics, the coordinate displacement of each feature point is found slightly and closely to neighboring timing interval. Therefore, we cut off a section of feature points in a fire profile by its leaning angle which is determined through the slope of the fire peak and the bottom point and stored it to another file called leaning state data. Then a state machine used for describing the fire transition state can receive the external force which is inputted by a user and is transformed to leaning degree of fire. The leaning angle is used to select the corresponding feature data set in the leaning state data. The advantage of this process not only can use resource efficiently but also change the fire status clearly. The relationship between input wind force and the leaning angle of fire can be found in [23].

We roughly divide the leaning degree of fire into six sets from -30 to 30 angles with 10 as interval, otherwise manually extract and classify the data in fire profile by leaning degree and storing to the leaning state data. The leaning state data consist of 12 sets of feature points that each set is captured in one period of time and it needs to be prepared before the external force entered. Due to the locality of feature points, we could approximately generate new ones of the next frame by fetching the data from relating leaning state back and forth. That means if the input force transforms into -20 degree, the state machine will jump to the leaning state which contains the fire feature points extracted at the leaning angle is between -20 and -30.

Fig. 7(a) illustrates the processing steps from creating external force to produce the corresponding leaning state for rendering. The external force entered by a user will be converted into the leaning angle of fire and triggered off the state transition in fire status state machine. The input of wind speed is equally separated into six levels from zero to five, the speed is increased or decrease by adding the current input. Fig. 7(b) shows the state machine for controlling the fire status by input force. The fire status state machine is designed for describing the leaning state of fire: The larger force, the larger leaning angle. But if the force sustains in a consistent input amount, the fire will stay in a flickering situation until the input changes.

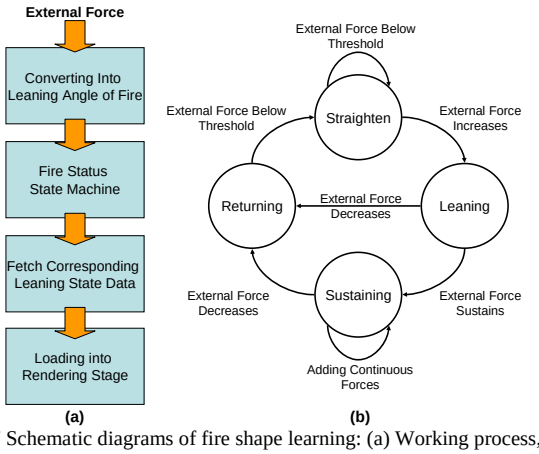


Fig. 7 Schematic diagrams of fire shape learning: (a) Working process, and (b) The fire status state machine

As mentioned in [24], fire has unique visual signatures. Color, geometry, and motion of fire region are all essential information for fire recognition. A region of fire can be defined in terms of (1) spectral characteristics, and (2) the spatial structure defined by their spectral variation.

The pixels in a fire region have different color spectra depending on its spatial locations. For color image, we might see the bright white color in the core, and yellow, orange and red away from the core to outer region. For grayscale images, we notice that the brightness is gradually decreasing from core to the periphery.

Here we simply divide the fire height and width into five by five grids and record the color value in each grid into spatial fire color correlation table (SFCCT). After processing the image samples, the SFCCT that consists of statistic information of every length and width section of flame, is also ready. Then it can be used as a lookup table for the rendering stage.

C. Rendering with a Particle System

The objective of this phase is to continually produce an acceptable and plausible visual effect. Here we use the

feature data and SFCCT that are created by interacting the fire status state machine with the external force and collocated with the particle system to generate fire simulation. This subsection focuses on describing how the particles vary from frame to frame. For describing fire effect by the particle system appropriately, appearance, color, shape, and behavior of diffusion are essential factors for rendering flame. These factors can be fulfilled well through the following two modules: region detector and path builder.

The processing flow of the rendering stage is shown in Fig. 8. First the external force is sent into the feature statistic learner, and the learner will select corresponding feature points from the leaning state data. And the particle system will calculate the position of each particle by path builder with the feature point set decided by fire status state machine and the dynamics the color of particles by region detector with SFCCT.

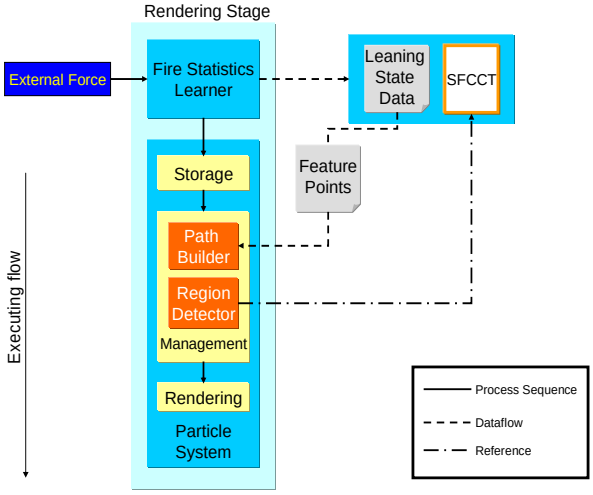


Fig. 8 The processing flow of the rendering stage

The objective of path builder is to arrange the path of particles inside the contour of fire. In the proposed method, we divide this work into two phases: The first phase is how the emitting area to be arranged; the second phase are how these particles to be shifted. On the other hand, the core idea of region detector is to change the color of particle depending on what grid of SFCCT it belongs. After generating results from two cross-section views independently, next step is to expand these isolated ones into a complete fire effect.

When implementing the particle system, all of the particles are supposed to be moved in one direction (e.g. upward). As illustrated in Fig. 9, all the newly born particles have assigned the lifespan parameter and the life will decrease gradually until back to zero.

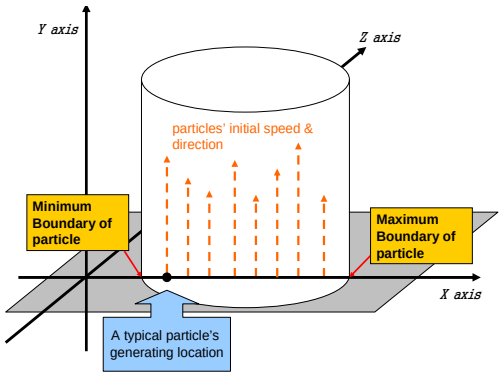


Fig. 9 The dynamic mechanism of a typical particle system.

However, the actual flame has its own color with light and shade. In order to represent the phenomena, all the particles have to change its color according to the spatial relationship of the current image. As illustrated in Fig. 10, the coordinates of particles must be traced to see if it moves from one color section to another in each frame. As a result, the transition of particles smoothly grows not only the position but also the hue of flame, based on the actual flame.

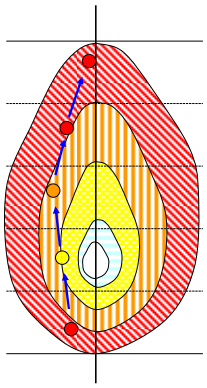


Fig. 10 Color of particles dynamic according to its region

The locus of particle is commonly shifted in a straight-line fashion. However, for the most part of fire generally corresponds with the crooked shape. Based on this feature, modeling the movement of a particle should be considered carefully. In the proposed method, the bottom area and the moving trace of particles are totally formed by the cubic spline to fit with the original image contour. The emitting position indicates the bottom boundary of fire which will be mapped out by the cubic curve form. Using the extremely points in previous processing step, one can formulate the bottom shape for particles to start off.

Since the emitting position is ready for the particle system, how these particles are shifted in the progression of fire simulation. It needs to provide the physical property for describing or controlling the velocity over time of an object moving from A to B. For example, an icon might follow a

parabolic curve in moving from A to B, rather than simply moving in a straight line fashion. From this perspective, there are huge advantages to use parametric cubic curves because the required data are points that can be obtained and manipulated despite lacking of other information like derivations and outer control points [25]. Now particles can move along the curve from the bottom point to the top point as shown in Fig. 11.

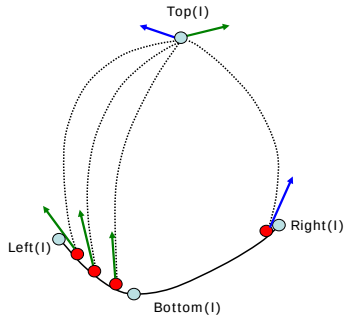


Fig. 11 Schematic diagram of particles' movement

So far we introduce the methods for extracting features from image and rendering 2D fire animation. For most applications, it is necessary to form a 3D fire animation. If two perpendicular views can be captured by two cameras that are perpendicular to each other, we can reconstruct 2D peripheral curve by connecting the feature points of these two planes by the cubic spline. Let the bottom point of these two planes be mapped onto the same world coordinate position. Therefore, when we look down the fire source, the peripheral curve will intersect the bottom point in a vertical fashion as shown in Fig. 12. F_1 and F_2 are used to denote the two sets of contour points in x-y plane and y-z plane respectively, and then Left(F_1), Right(F_1), and Bottom(F_1) denote the extremely left, right, and bottom points in the image captured from x-y plane , respectively. In the proposed method, we don't force to model the outer appearance but intersect several supplementing slices between two orthographical images.

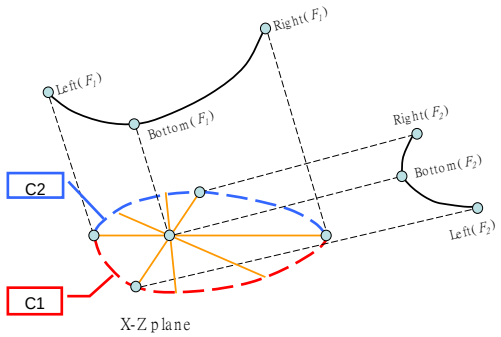


Fig. 12 Using feature points to form a circular region by cubic-spline

IV. IMPLEMENTATION

In this section, implementation of fire simulation and the rendering result are discussed. The experiments were run in the PC environment, PC with an AMD Phenom FX-5000 and 4 Gigabytes main memory. The implementation is divided into two parts based on two main stages in the flowchart. The preprocessing stage is mainly to process the image data in offline. After preprocessing, one can obtain features from images and collect them into a fire profile. Based on the fire profile, one may generate realistic fire image. To use resource more efficient, the feature statistics learner is used to collect fire dynamics information through the fire profile. The procedures of the whole system are given as follows:

- Using video editing tool to separate a video into image sequences.
- Using thresholding and the Sobel detector to generate contour map.
- Extracting and gathering image features into a fire profile for rendering.
- Analyzing fire profile to build up the framework of fire dynamics.
- Generating fire simulation.

We use Microsoft Visual Studio C# 2005 as the development tool and the JSEG method [18] to develop an image processing tool for users to easily accomplish the image processing work. Several image processing skills are integrated to the program that can be used by one-click interface as shown in Fig. 13. Image functions include "ColorInvert", "Edge Detection", and "Segmentation." Users can press the option "MenuOn" of "JSEG" to open parameter panel "JSEGform." "JSEGform" provides some control options, such as the threshold for quantization, display, and merge.

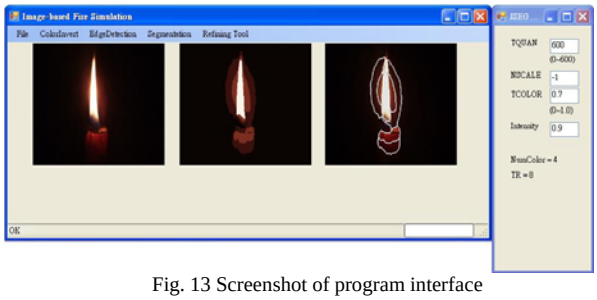


Fig. 13 Screenshot of program interface

The 3D simulation result is through combining with the different 2D results of the proposed algorithm. Fig. 14 shows the three dimensional distributions of particles. As the particles are independent objects, one can receive the outcome that can be calculated with virtual environment about the collision or the confliction. These particles are attached with a texture to formulate like a cluster of particles in the final representation.

Fig. 15 illustrates an example for showing how the particles to be rendered. A particle can be expanded into a quadrangle and regarded as a display primitive. Finally the display primitives seems like a spreading fire effect by attaching a texture on each of it and the blending effect can smooth the edge of these overlapping polygons. Fig. 16 shows the FPS measurements for the different slice counts. These slices would be circularly placed inside the fire volume and the particle number is 100 in each slice.

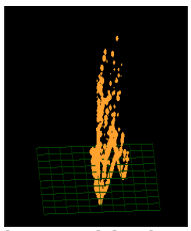


Fig. 14 Three dimensional distributions of particles

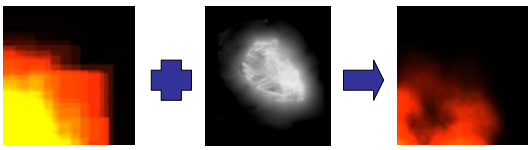


Fig. 15 Example of billboarding

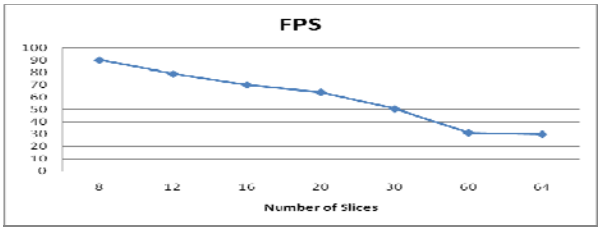
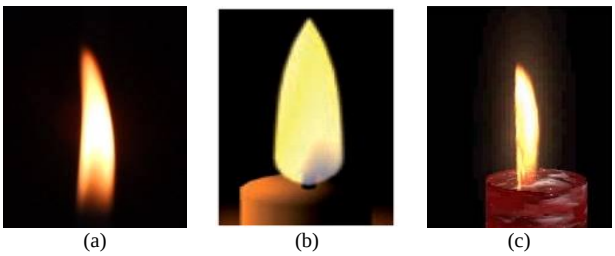


Fig. 16 FPS measurements for the different slice counts

Fig. 17 is comparisons of real candle light video with the rendering result created by the proposed method and the one in [26]. In contrast to Fig. 17(a), the image result of our implementation Fig. 17(c) is more realistic than that of Fig. 17(b).



(a) (b) (c)

Fig. 17 Comparisons of the simulation results : (a) real image (b) implemented by Bridault-Louchez et al. [26], and (c) the proposed method

V. CONCLUSIONS

In this paper, we have presented a visual learning method for simulating fire phenomenon through fire video. The proposed method provides visual reference for the expecting ability of simulating result and enhances the reality of rendering effect indeed. Nevertheless, the proposed method has the learning mechanism which can generate continuous animation without the restriction of a short clip of film and allow to receive input as external force to change the leaning status of fire.

Therefore, the proposed algorithm can not only eliminate the heuristic property of the procedural method but also provide a more flexible method than other reconstructing work. And the generated fire animation is not only realistic but also convenient for embedding to various existing virtual scenes.

REFERENCES

[1] S. Hasinoff and K. Kutulakos, “Photo-consistent reconstruction of semi-transparent scenes by density sheet decomposition.” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 5, pp. 870–885, 2007.

[2] I. Ihrke and M. Magnor, “Adaptive grid optical tomography,” Graphical Models, Vol. 68, No. 5, pp. 484–495, 2006.

[3] N. Chiba, K. Muraoka, H. Takahashi, and M. Miura, “Two-dimensional visual simulation of flames, smoke and the spread of fire,” The Journal of Visualization and Computer Animation, Vol. 5, No. 1, pp. 37–54, Jan.–Mar, 1994.

[4] W. T. Reeves, “Particle Systems – A technique for modeling a class of fuzzy objects,” ACM Transactions on Graphics, Vol. 2, No. 2, pp. 91-108, April 1983.

[5] A. R. Fuller, H. Krishnan, K. Mahrous, B. Hamann, and K. I. Joy, “Real-time procedural volumetric fire,” In Proceedings of the 2007 symposium on Interactive 3D graphics and games , pp.: 175 – 180, 2007.

[6] Y. Takai, K. Ecchu, and N. K. Takai, “A cellular automaton model of particle motions and its applications,” The Visual Computer, Vol. 11, No. 5, pp. 240–252, 1995.

[7] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita, “A simple, efficient method for realistic animation of clouds,” In Proc. ACM SIGGRAPH 2000, pp. 19–28, 2000.

[8] X. Wei, W. Li, K. Mueller, and A. Kaufman, ”Simulating fire with texture splats,” In Proc. IEEE Visualization, pp. 227-237, Nov. 2002.

[9] M. Balci, M. Alnasser, and H. Foroosh, "Image-based simulation of gaseous material," Image Processing, 2006 IEEE International Conference on, pp. 489-492, Oct. 2006.

[10] D. Nguyen, R. Fedkiw, and H. Jensen, “Physically based modeling and animation of fire,” In Proc. ACM SIGGRAPH 2002, pp. 721–728, 2002.

[11] J. Hong, T. Shinar, R. Fedkiw, “Wrinkled flames and cellular patterns,” ACM Transactions on Graphics, Vol. 26, No. 3, Article 47, July 2007.

[12] J. Stam and E. Fiume, “Depicting fire and other gaseous phenomena using diffusion processes,” In Proc. ACM SIGGRAPH 1995, pp. 129–136, 1995.

[13] P. S. McCormick and J. P. Ahrens, “Visualization of wildfire simulations,” IEEE Computer Graphics and Applications, Vol. 18, No. 2, pp. 17–19, Mar.–Apr. 1998.

[14] R. W. Bukowski and C. H. Séquin, “Interactive simulation of fire in virtual building environments,” In Proc. ACM SIGGRAPH 1997, pp. 35–44, 1997.
[15] H. Rushmeier, A. Hamins, and M. Y. Choi, “Volume rendering of pool fire data,” IEEE Computer Graphics and Applications, Vol. 15, No. 4, pp. 62–66, July 1995.
[16] D. Laur and P. Hanarahan, “Hierarchical splatting: a progressive refinement algorithm for volume rendering,” In Proc. ACM SIGGRAPH 1991, pp. 285–288, 1991.
[17] J. Stam, “Stochastic rendering of density fields,” In Proc. Graphics Interface 1994, pp. 51–58, 1994.
[18] T. McReynolds and D. Blythe, “Advanced Graphics Programming Techniques Using OpenGL,” Course presented at ACM SIGGRAPH 1999, available at http://www.opengl.org/developers/code/sig99.
[19] S. A. King, R. A. Crawfis, and W. Reid, “Fast volume rendering and animation of amorphous phenomena,” In Proc. International Workshop on Volume Graphics 1999, pp. 229–242, 1999.
[20] J. Takahashi, H. Takahashi, and N. Chiba, “Image synthesis of flickering scenes including simulated flames,” IEICE Transactions on Information and Systems, E80-D(11):1102–1108, Nov. 1997.
[21] G. Cox and R. Chitty, “Some source-dependent effects of unbounded fires,” Combustion and Flame, Vol. 60, No. 3, pp. 219-232, 1985.
[22] B J. McCaffrey, “Purely buoyant diffusion flames: some experimental results,” National Bureau of Standards, NBSIR 79-1910, 1979.
[23] P. P. K. Raj, A. N. Moussa, and K. Aravamudan, “Experiments involving pool and vapour fires from spills of liquefied natural gas on water,” US Coast Guard Report, No. CG-D-55-79, 1979.
[24] C. B. Liu and N. Ahuja, “Vision Based Fire Detection,” In Proc. of Int. Conf. on Pattern Recognition, Vol. 4, 2004.
[25] Edward Angel, “Interactive Computer Graphics: A Top-Down Approach Using OpenGL,” Addison-Wesley, Fifth Edition, 2009.
[26] F. Bridault-Louchez, M. Leblond, F. Rouselle, and C. Renaud, “Real-time rendering and animation of plentiful flames,” In Proc. of the 3rd Eurographics Workshop on Natural Phenomena, 2007.