IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 14, 2019, accepted August 4, 2019, date of publication August 9, 2019, date of current version August 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2934163

Fire Sprite Animation Using Fire-Flake Texture and Artificial Motion Blur

JONG-HYUN KIM[1] **AND JUNG LEE**[2]
[1]Division of Software Application, Kangnam University, Yongin 16979, South Korea
[2]School of Software, Hallym University, Chuncheon 24252, South Korea

Corresponding author: Jung Lee (airjung@hallym.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2017R1C1B5074984), and in part by a Hallym University Research Fund (HRF-201811-013).

ABSTRACT In this paper, we propose a sprite animation synthesis technique that can efficiently represent the fire-flake effects seen in the natural phenomenon of fire. The proposed method uses the actual fire video or animated fire video as inputs and performs the following steps: 1) Extraction of feature vectors that can predict the direction of flame from image, 2) calculation of artificial buoyancy field, 3) creation and advection of fire-flake texture, 4) calculation of artificial motion blur using buoyancy flow, and 5) high quality composition. First, we detect the edges from the image and calculate the feature vectors needed to calculate the artificial buoyancy field. The computed 2D feature vectors are integrated into the Navier-Stokes equation and used to calculate the buoyancy field, which generates and advects anisotropic fire-flake textures. Finally, we apply artificial motion blur according to buoyancy direction to improve composition result of sprite animation. As a result, this method is based on image synthesis, which is faster than the existing 3D simulation-based approach. Experimental results show that high quality results can be easily and reliably obtained. In addition, since the final result is a sprite animation format, it can be easily used in existing game engines.

INDEX TERMS Sprite animation, fire effects, anisotropic fire-flake texture, artificial buoyancy field.

I. INTRODUCTION

Fire visual effects are often used in movies, animations, games, and image processing. Professional designers fascinatingly express fire effects by adjusting the color and style of the fires in the images or videos. In recent years, fire effects have been inserted into movies and games using fire style templates or assets built into the game engine (see Figure 1).

Sprite animation is a commonly used method for expressing fire effects, which is often used in game engines to synthesize fire effects by copying and pasting a fire template image into an original images or videos. Professional designers manually edit and express detailed attributes such as shape and direction of fire by cutting, pasting, deleting, rotating, and resizing fire images in layer style. This method can produce stable and easy results. However, due to the inherent limitations of the designer's manual work, it is very difficult to express the shape and movement of realistic fire, and a highly skilled designer must work over a long period of time.

(a) (b)

FIGURE 1. Various fire effects in animation and game (a: animation 'ZETMAN', b: mobile game 'Raising Fire Magician'.

The fire effect based on a still frame can be automatically generated using a procedural approach [1], [13]. However, the procedural approach makes it difficult to add or edit effects other than predefined fire effects. A simulation-based approach can produce realistic and very detailed flame effects. The shape and flow of the flame can be intuitively controlled by placing the fuel to follow the target shape, burning it, and then inducing the flame in the desired direction. However, because of the large amount of computation, simulating fire simulation in low-resolution is required to

The associate editor coordinating the review of this article and approving it for publication was Lefei Zhang.

perform interactive simulations, and as a result, detail of fire is degraded. In other words, high-resolution simulation is required to produce high-quality fire effects, which are not suitable for interactive systems because of the high computational complexity.

An important factor when using simulation in video special effects is the balance of quality and speed, and often they are in opposition to each other. In this paper, we propose a new framework that can efficiently represent high quality fire effects by analyzing images of real fire and high resolution simulation images. The proposed method analyzes the input image to compute artificial buoyancy, and uses this field to generate and advect fire-flake textures. Each step is calculated in a separate layer, and the final result is made through composition of all layers. As we analyze and import realistic shape changes and movements of fire from input video, our technique can produce high quality fire sprite effects.

II. RELATED WORK

Fluid simulations are a demanding area for a variety of special effects such as movies, games, and art. Stam proposed a semi-Lagrangian technique to improve the numerical stability of fluid simulations [14]. Foster and Fedkiw used the semi-Lagrangian method and the conjugate gradient method, a system that effectively solves linear equations, to calculate the motion of water, and used a levelset to represent surfaces of water [15]. Enright et al. have improved the accuracy of fluid simulation by proposing a particle-levelset method that minimizes the amount of fluid volume lost during the simulation [7].

In the field of physically based simulation, smoothed particle hydrodynamics (SPH) method, which is a technique to efficiently calculate particle-based water simulation, is also widely used [16]. Becker and Teschner proposed a weakly compressible SPH (WCSPH) technique that uses a state equation to reduce the computational complexity of the SPH technique [12]. Solenthaler et al. then proposed Predictive Corrective Incompressible SPH (PCISPH), which uses the prediction-correction technique to calculate the pressure of a particle to ensure the incompressibility of the fluid [11]. This technique predicts the density from the velocity and position of the particle before computing the pressure in the simulation process, and iteratively calculates the incompressible pressure based on the predicted density value. Macklin and Müller [17] used a position-based constraint between neighboring particles to satisfy the incompressibility of the fluid [18], and the algorithm was simpler than PCISPH and stably performed at larger time-steps.

Nguyen proposed a level-based fire simulation and rendering technique using incompressible fluids [10]. This technique uses fuel, flame, and post-combustion media, and realistically expresses the motion of fire using the velocity before and after combustion. Hong combines the detonation shock dynamics (DSD) technique with the Navier-Stokes equation to detail the flame wrinkled patterns [4]. Horvath and Geiger proposed a framework for accelerating hybrid particle-grid simulations using multi-GPU [8]. This technique divides the simulation space into a number of 2D slices, and enables high-resolution fire simulation to be performed quickly. Kim et al. proposed a method of containing flame in a target shape by controlling the temperature [3]. In the field of fire simulation, a technique has been proposed to express the motion of fire-flake as well as flame according to the temperature field and velocity field of fluid [5]. The simulated fire-flake motion in this technique is not sufficient to represent realistic fire-flake effects because it is a random walk based, with somewhat noisy motion unlike flame.

In recent years, studies have been made to improve surface details of water simulation based on texture synthesis techniques. There have been attempts to utilize 2D texture synthesis techniques for simulation, but problems such as texture blurring or blending of colors have resulted in the reduction of detail in the advection process. Gagnon et al. [6] has improved the details of texture-based water surfaces by reducing these problems. In another aspect, research has been conducted to improve simula- tion efficiently using machine learning and artificial intelligence [19]–[22].

III. OUR FRAMEWORK

The algorithm of the proposed method is performed in the following order after receiving the sprite image sequence.

1) Calculation of feature vectors to extract the motion of the flame from the input images.
2) Computation of the buoyancy field based on feature vectors : This process improves the details of the buoyancy field that changes over time by including buoyancy throughout the advection process as well as the current time.
3) Creation of anisotropic fire-flake texture and its advection according to buoyancy flow.
4) Calculation of the artificial motion blur kernel based on the direction of buoyancy, and improving the quality of the final composition result using the kernel.

A. EXTRACTION OF FEATURE VECTORS FROM IMAGE SEQUENCES

To calculate the feature vectors Fv from the input image, its edges are first extracted and the final feature vector Fv^* is calculated using the derivative of the extracted edges. The overview of the calculation process is shown in Figure 2.

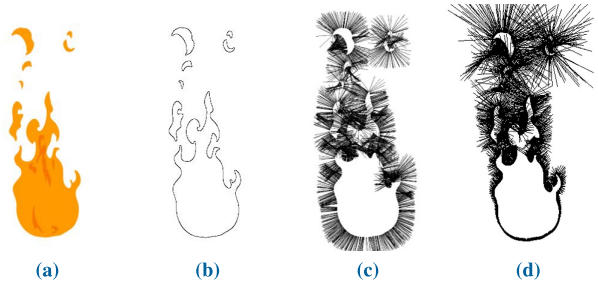


FIGURE 2. Overview of feature vector calculation : (a) one of input image sequences, (b) extracted edges, (c) calculated feature vectors *Fv*, (d) final refined feature vectors *Fv**.

First, edges are extracted by using a difference of Gaussian (DoG) filter based on the second derivative to perform edge detection. This filter assigns a different variance value to each Gaussian operation and calculates the edge map using the difference of the results. The equation is as follows (see Equation 1).

$$f^{DoG}(x,y) = G_{\sigma_1} - G_{\sigma_2} = \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1}e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2}e^{-(x^2+y^2)/2\sigma_2^2}\right) \tag{1}$$

The edge thickness to be detected can be adjusted by changing the values of σ_1 and σ_2 in this equation. The direction of the artificial buoyancy is calculated using the gradient of the detected edges (see Equation 2).

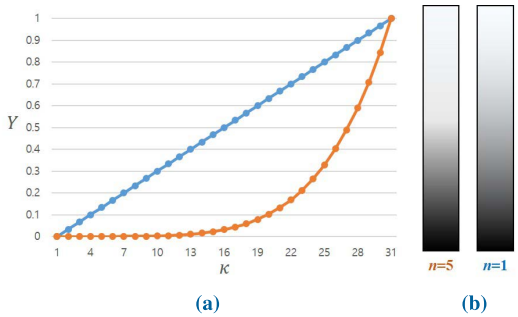
$$\nabla f^{DoF}\kappa = \underbrace{\left(\frac{\partial f^{DoG}}{\partial x}, \frac{\partial f^{DoG}}{\partial y}\right)}_{\nabla f^{DoG}}\underbrace{\left(\frac{w^y}{w^{height}}\right)^n}_{\kappa} \tag{2}$$

κ is a variable that artificially controls the magnitude of buoyancy. To extract Fv with motion of large variation, we calculate Equation 2 only for the region where the following condition is satisfied : $\|\nabla f^{DoF}\| > \varepsilon$, which is set to 40 in this study. Generally, since the turbulence flow in the fire effects is strong after combustion, the strength of the turbulence is controlled using κ based on the Y-axis. In actual fire, a larger turbulent flow is seen when the flame moves upward due to buoyancy than when the flame is generated at the emission position (see Figure 3). Since this flow makes the motion of the fire-flake more complicated, in this study, the kappa is adjusted so that the feature vector becomes larger as the Y coordinate value becomes larger. In Equation 2, w^y is the y coordinate of the image and w^{height} is the height of the input image. As a result, the κ value is close to zero at the bottom, and approaches one at the top. n is a variable for controlling the size of the Fv. As shown in the Figure 4, we set n to 5 and use the weight function of the curve type so that the difference of the Fv becomes larger as the size of n increases.

(a) (b)

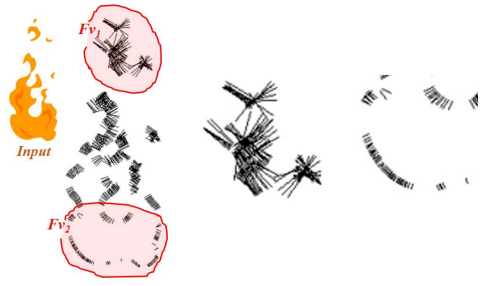
FIGURE 3. Real sparks and flakes of fire.

The Figure 5 shows the calculated Fv according to κ and n. As shown in the figure, the upper region after burning has a much larger Fv than the lower region where the fuel is



(a) (b)

FIGURE 4. Changes in κ with different n values. (a) Weight curves according to different n values (orange : 5, blue : 1), (b) Color bars according to different weight graphs.



(a)Scaled Fv (b) (c)

FIGURE 5. Controlling Fv by κ.

generated. If there is no κ, buoyancy is the same everywhere, so fire-flakes may move downwards. (see Figure 2c). Although Fv is controlled by κ, the direction of Fv may be different from that of buoyancy because it has a first derivative with respect to the edge, so it looks like a normal vector (see Figure 2c). In this paper, Fv is refined based on the default buoyancy force $F_{buoyancy}$ to calculate the artifcial buoyancy field according to the direction and size of Fv (see Equation 3).

$$Fv^* = F_{buoyancy}\left\|\nabla f^{Dof}\kappa\right\|\beta + \nabla f^{Dof}\kappa \tag{3}$$

$F_{buoyancy}$ is a 2D vector whose range is [0,1] and β is set to 0.02 in this paper. As a result, Fv^* is expressed as a force that moves according to the direction of the flame (see Figure 6).

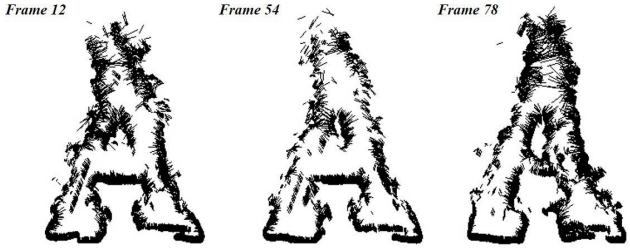


FIGURE 6. Visualization of Fv^* by frames.

B. GENERATION OF BUOYANCY FIELD

In this paper, we model the buoyancy field that changes with time by integrating Fv^* and Navier-Stokes equations to express the natural motion of fire-flakes. First, to compute the differential form of the momentum equation representing the

motion of a fluid, we apply Newton's second law to a fluid particle with mass dm (see Equation 4).

$$F = \frac{P}{dt} \tag{4}$$

The linear momentum P is calculated as follows (see Equation 5).

$$P = \int_{mass} udm \tag{5}$$

Newton's second law for mass dm in the infinitesimal system can be written as Equation 6.

$$dF = dm\frac{du}{dt} \tag{6}$$

After obtaining the acceleration equation from the velocity field, we can rewrite Newton's second law as vector equation as follows (see Equation 7).

$$dm\frac{Du}{Dt} = dm(\frac{\partial u}{\partial t} + u \cdot u) = dF \tag{7}$$

Here, the influence of mass dm and volume $dV = dxdydz$ on the differential element must be considered when calculating dF. Generally, dF in fluid mechanics is composed of surface force and body force as follows (see Equation 8).

$$dF = dF_{surface} + dF_{body} \tag{8}$$

Surface forces acting on the surface can be rewritten using viscous force and pressure difference as follows (see Equation 9).

$$dF_{surface} = \{\mu\nabla \cdot (\nabla u) - \nabla p\}\, dV \tag{9}$$

Body forces are the forces exerted on the whole body of a finite element. In the classic Navier-Stokes equation, gravity is the only body force acting on a fluid. Instead of gravity, we can use the buoyancy described above to rewrite the body force as follows (see Equation 10).

$$dF_{body} = U_{potential}dV \tag{10}$$

Finally, the body force is calculated as follows (see Equation 11).

$$U_{potential} = U_{gravity} + U_{buoyancy} \tag{11}$$

Here $U_{buoyancy}$ is a user defined artificial buoyancy, which can be used to rewrite Navier-Stokes equations as follows (see Equation 12).

$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho uu) = \mu\nabla \cdot (\nabla u) - \nabla P + \rho g + U_{wind} \tag{12}$$

The only difference between Equation 12 and the general Navier-Stokes equation is the existence of $U_{buoyancy}$, a user defined buoyancy. This buoyancy field is computed from the input images and does not have a large amount of computation and is easily combined with existing fluid equations as seen above.

In the Figure 7a, Fv_1 has a relatively large value compared to Fv_2, and our method stably calculates the buoyancy field by

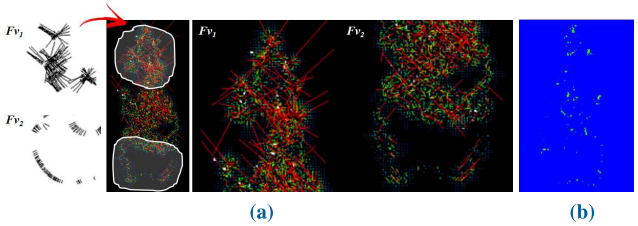


FIGURE 7. Calculating artificial buoyancy field and candidate position of fire-flake texture : (a) buoyancy field by Fv^*, (b) candidate positions of fire-flake textures.

automatically adjusting the size of Fv. The Figure 7b shows the candidate position C_{pos} where the fire-flake texture is to be created, where the size of its Fv^* is larger than the user-defined threshold value : $\|Fv^*\| > \alpha$, where alpha is set to 0.03 in this paper.

Extracting buoyancy from images without using two parameters, κ and n, presented when refining Fv^*, often results in fields containing errors (see Figure 8). In physics-based simulation, it is rare that errors accumulate because velocity is recalculated from the physics equation every frame, but when analyzing from image, a new problem arises that is not present in the simulation approach. As mentioned earlier, the gradients calculated from the edges (see Figure 2b) are mostly similar in size to each other(see Figure 8a). This feature does not disappear even with the addition of the advection process, the buoyancy field flows downward as shown in the Figure 8b. The following section describes in detail how to generate fire-flake textures in candidate positions and advect them.

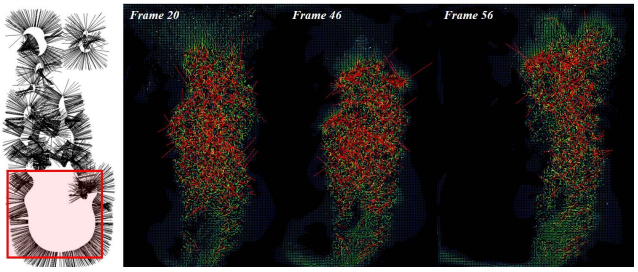


FIGURE 8. Wrong buoyancy field by cumulative error when using Fv without κ.

C. GENERATION OF ANISOTROPIC FIRE-FLAKE TEXTURES

In this paper, texture is created at C_{pos} to represent fire-flake. Texture is created in circle form at C_{pos}, and it is scaled and rotated with a vector $u_{C_{pos}}$ sampled from the buoyancy field Bv (see Figure 9). Bilinear interpolation was used to calculate $u_{C_{pos}}$, and the color of fire-flake was determined according to $\left\|u_{C_{pos}}\right\|$. Anisotropy of fire-flake texture is also calculated using $u_{C_{pos}}$. In addition, to compute the detailed motion of the fire-flake effects, we apply the jitter value τ to the velocity of the fire-flake particle using the stochastic solver used in the dispersed bubble flow [2] (see Equation 13).

$$\tau = cos\theta = (2\xi + k - 1)/(2k\xi - k + 1) \tag{13}$$

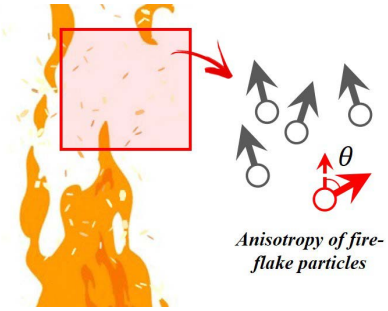


FIGURE 9. Overview of fire-flake flow. θ indicates the altered direction (color table of fire-flakes:).

where $\theta, \xi \in [0, 1]$, and $k \in [-1, 1]$ represent the altered direction, a uniform random number, and the scattering coefficient, respectively. Here, $k = 0$ gives isotropic scattering, $k > 0$ is forward scattering, and $k < 0$ is backward scattering.

D. ARTIFICIAL MOTION BLUR

After the generation of fire-flake texture, we apply buoyancy field-based motion blur for high-quality composition. The basic principle of motion blur is to add up the radiance contributions over time, which can be expressed as

$$L_p = \int_{t_s} \int_A L(x', \vec{\omega}, t) s(x', \vec{\omega}, t) g(x') dA(x') dt, \qquad (14)$$

where g is the filter function, s is the shutter exposure, L is the radiance contribution from the ray [9]. The above principle applies to both Lagrangian and Eulerian motion blurs. In Equation 14, x is the place where the movement of objects jumps into the motion blur; for the evaluation of x', the locations of the objects at arbitrary moments need to esimated, which forms a core part of motion blur.

The Lagrangian motion blur technique is commonly used to render explicit surfaces such as rigid bodies, deformable solids, and clothes. The core part of the Lagrangian motion blur approach is to compute ray-object intersection at arbitrary super-sampled instants from the given 3D data of each frame. In Lagrangian motion blur, the estimation is done by taking the time-interpolation of the vertices of the two involved frames. When the position $x(t^n)$ and $x(t^{n+1})$ of the vertices at t^n and t^{n+1} are given, the estimated position $x_L(\delta)$ at super-sampled time δ is calculated by

$$x_L(\delta) = \frac{\delta - t^n}{t^{n+1} - t^n} x(t^{n+1}) + \frac{t^{n+1} - \delta}{t^{n+1} - t^n} x(t^n). \qquad (15)$$

We now briefly consider the physical meaning of the estimation given by rearranging Equation 15 into the form

$$x_L(\delta) = x(t^n) + (\delta - t^n) \frac{x(t^{n+1}) - x(t^n)}{t^{n+1} - t^n}. \qquad (16)$$

This equation is the result of assuming that movement is made of constant velocity : $(x(t^{n+1}) - x(t^n))/(t^{n+1} - t^n)$.

In this paper, to improve the composition result, we exaggerate the motion of the fire effect and calculate the motion blur considering the buoyancy field Bv rather than the constant velocity (see Figure 10).

$$x_A(\delta) = x(t^n) + (\delta - t^n) Bv(t^n) \kappa. \qquad (17)$$

κ controls the intensity of the motion blur and is the weight used to model the Fv (see Equation 2). As a result, the direction of the velocity is varied by Bv for each position of the image space, and the magnitude of the velocity is variously scaled by κ (see Figure 10). In the proposed artificial motion blur, the blur effects are expressed more strongly by κ value, which becomes larger as it goes up, so that the fire-flake effects similar to the real ones can be expressed. Figure 11 compares the results before and after applying motion blur, and shows some more visually pleasing fire effects when motion blur is applied. κ is also used efficiently in the composition process, and our image-based framework produces motion blur effects similar to those using physics-based simulation techniques.

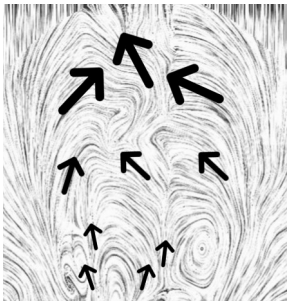


FIGURE 10. Motion vectors with sampled velocity on buoyancy field.

FIGURE 11. Comparison of artificial motion blur.

IV. RESULTS

In this paper, we compare 9 different scenarios to analyze the proposed fire sprite animation in various aspects. Various experiments based on real fire video, animated fire video, and sprite animation produced by the designer show that this method produces stable fire-flake effects.

Figures 12 and 13 show the result of experiment by inputting fire video produced by level-set simulation [4]. Fire-flake textures were generated and advected to flame motion without calculating Eulerian grid-based fluid simulation. Generally, plug-in type particle effects assets are awkward because they do not consider flame motion but simply paste particle effects by layer. But our results have produced

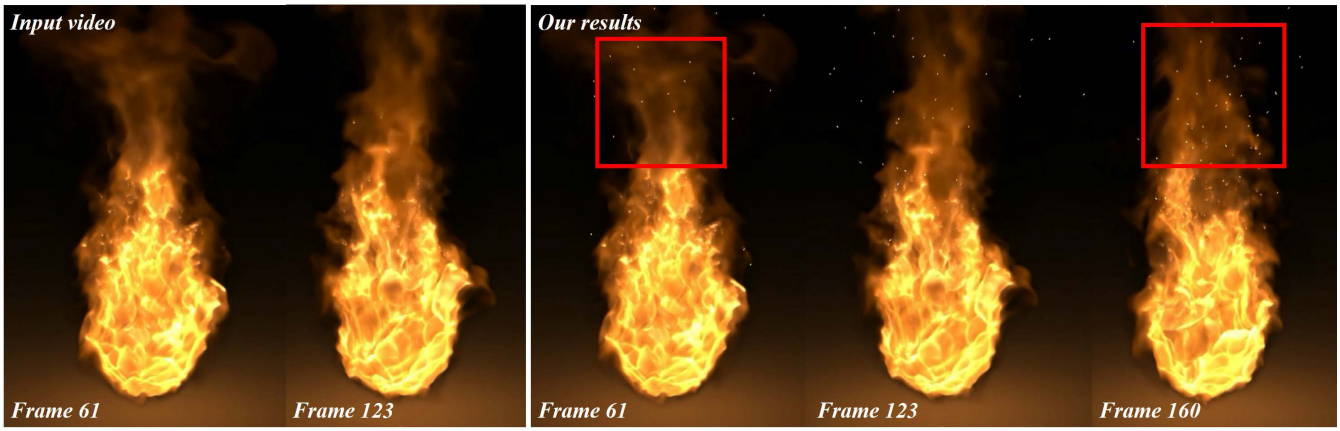


FIGURE 12. Fireball effects (input : simulated video by Hong et al. [4]).

FIGURE 13. Zoom-in results (red box in Figure 12).

fire-flake effects of natural motion as if obtained by flame simulation.

Figures 14 and 15 show the result of applying this method to actual fire video. As shown in the figure, fire-flake effects are generated in accordance with the flame direction, resulting in natural movement. In general, generating such fire-flake effects is more difficult and time-consuming to express than flame, because the number of particles to represent fire-flake is very high and the influence of flame motion on the fire-flake must be considered. However, since this method is an image-based framework, it can produce fire-flake animation results with good quality in a short period of time.

Figures 16∼18 are the results of experiment with animated images made by designer. Like the previous results, our method produces natural motion as calculated by the simulation approach, and fire-flake effects are synthesized naturally on the image space.

Figure 19 shows that the flame with an artificial shape produces a fire-flake effect well. Despite the small amount of computation, the proposed method has well extracted the feature vectors, fire-flake effects, buoyancy fields, etc.

We also applied our method to various cartoon style animated fire images (see Figure 20∼23). Because our technique is a 2D image-based framework, it is more flexible than the physics-based simulation approach without numerical instability, so we can easily produce all of these scenes using the same parameters except for Figure 23.

Figure 22 shows the results of adjusting the quantity of fire-flakes effects according to the different parameter values, α. The smaller the α value, the wider the region of the candidate position where the fire-flake texture is to be created, thereby increasing the quantity of fire-flake effects. Rather than merely increasing quantity, textures are advected by Bv, so the designer can flexibly use the appropriate fire-flake effects for the situation.

Unlike the previous results, Figure 23 shows that fire effects are expressed in a closed space such as a drum. In the

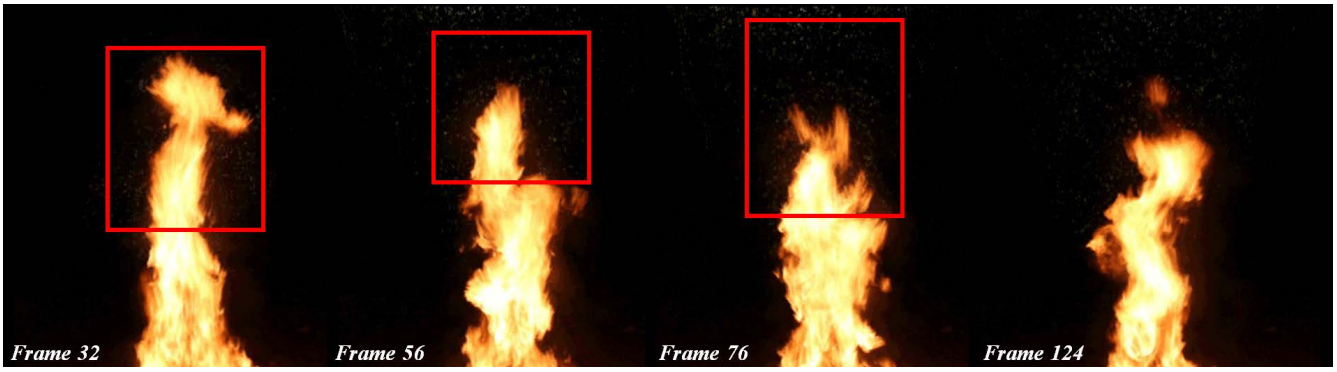


FIGURE 14. Campfire effects (input : real fire video).

FIGURE 15. Zoom-in results (red box in Figure 14).

FIGURE 16. Cartoon fire effects1.

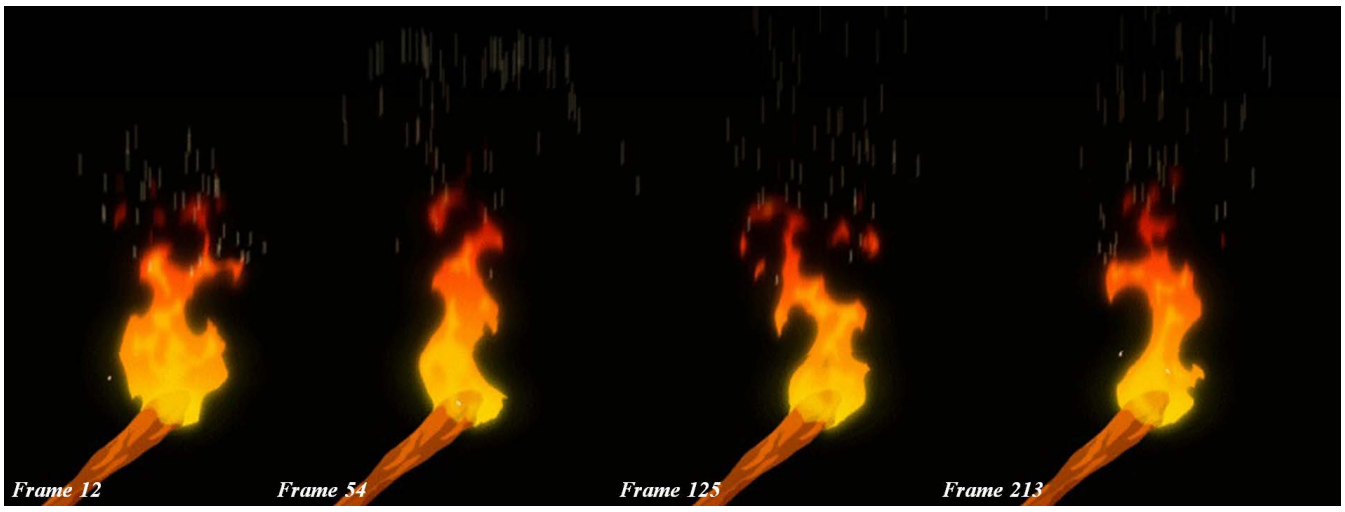


FIGURE 17. Cartoon torchlight effects.

process of calculating feature vectors, Fv^* is extracted not only from the flame but also from the drum. Therefore, since fire-flake textures are generated from the drum during the calculation of C_{pos}, the following clamping function was used in this scene : $\|Fv^*\| \leftarrow \|Fv^*\| < 2.0 \,?\, 0 : \|Fv^*\|$.

Generally, in order to express fire-flake effects in simulation approach, 3D grid-based simulation must be calculated and various parameters (grid resolution, time-step, viscosity, etc.) should be set. In addition, in order to get the desired result in this process, many trial and error must be done. However, the proposed method is much faster and stable than the simulation approach because it uses the motion analyzed from the fire video which is already made with high-quality. In addition, it is possible to animate more frames than the number of input images, and 2D simulation technique is added in the process of creating Bv, so that a natural buoyancy field can be created even if the number of input images is small. The table summarizes the environment and performance for the results presented in this paper.

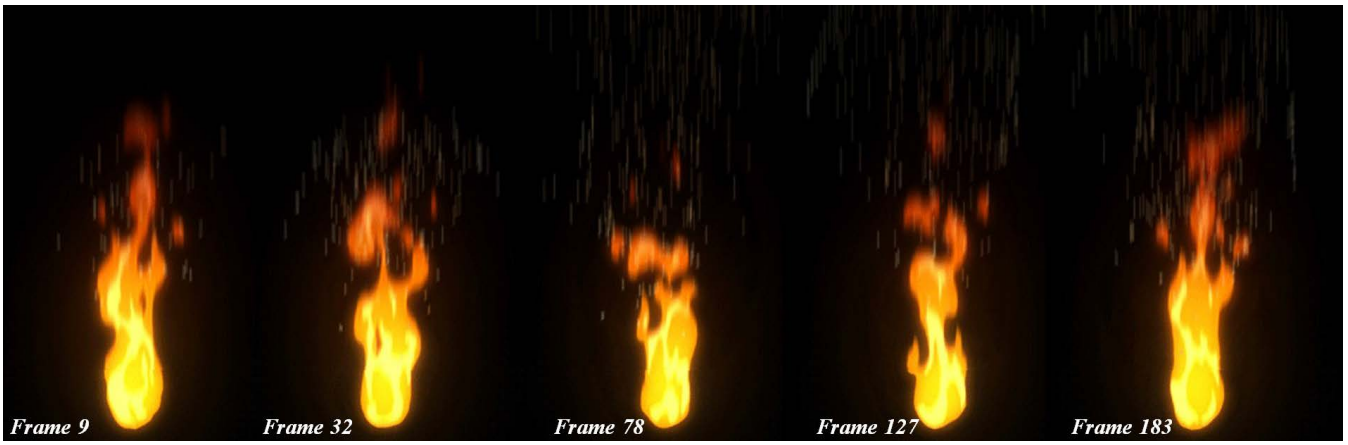
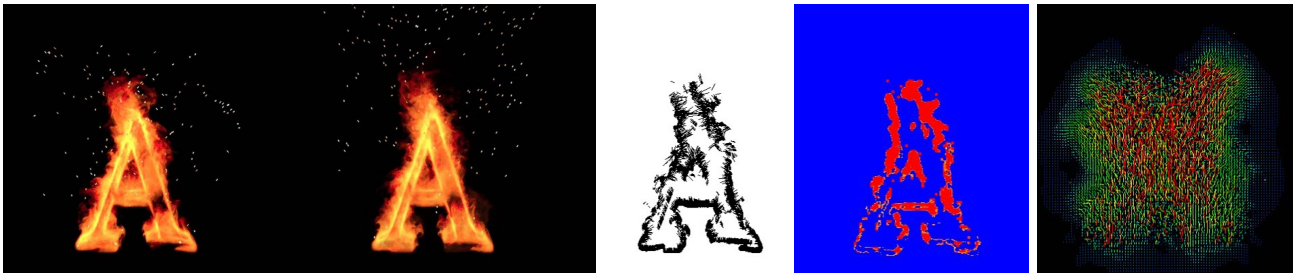
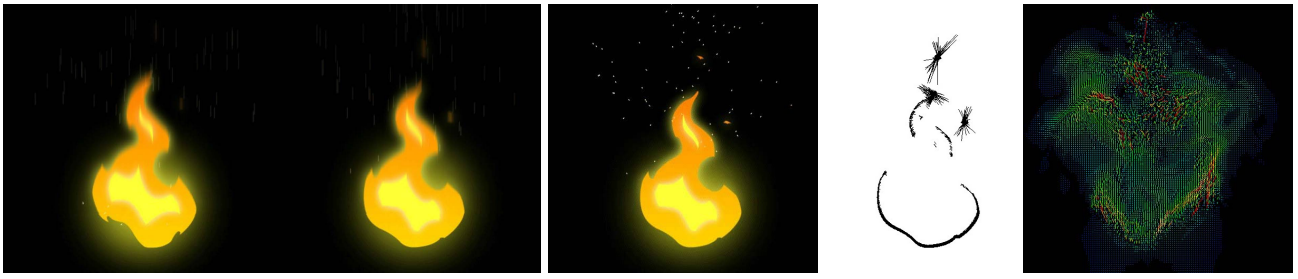


FIGURE 18. Cartoon fireball effects.



(a)Without motion blur (b)Fv^* (c)C_{pos} (d)Bv

FIGURE 19. Fire-flake effects generated in the A-shaped flame.



(a)Final results (b)Without motion blur (c)Fv^* (d)Bv

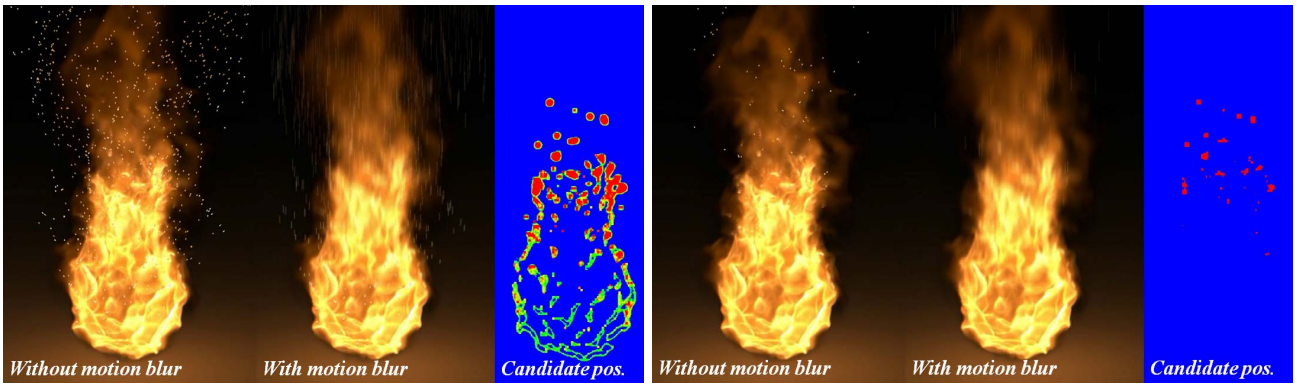
FIGURE 20. Cartoon fire effects 2.

(a)Final results (b)Without motion blur (c)Fv^* (d)Bv

FIGURE 21. Cartoon campfire effects.

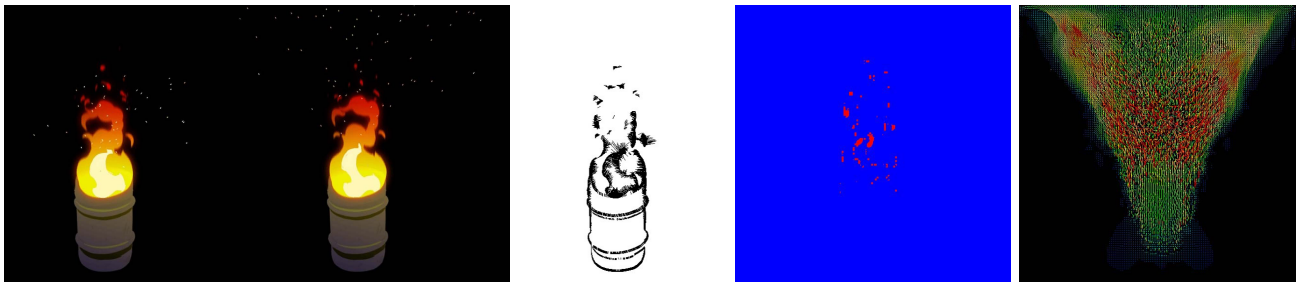
V. DISCUSSION

In this paper, we propose a novel framework to efficiently capture high quality fire flake effects by receiving videos. The fire flake effects required in the CG/VFX industry are often expressed as secondary effects of fire, and are generally calculated based on the motion of the underlying



(a)α : 0.005 (b)α : 0.03

FIGURE 22. Controlling of the quantity of fire-flake effects with differents parameters.



(a)Without artificial motion blur (b)Fv^* (c)C_{pos} (d)Bv

FIGURE 23. Cartoon fire effect3.

TABLE 1. Computation time and size of our example scenes (avg. num. : average number).

Figure	Num. of input images	Size of input image	Grid res. of Bv	Avg. num. of fire-flake textures	Avg. computation time (frames/sec)
12	299	640×640	160×160	58	0.42
14	13	391×391	98×98	8,519	0.14
16	42	400×400	100×100	543	0.12
17	23	500×500	125×125	127	0.18
18	34	415×415	104×104	321	0.12
19	106	500×500	125×125	210	0.18
20	16	512×512	128×128	95	0.21
21	32	600×600	150×150	182	0.37
23	14	600×600	150×150	87	0.62

fire simulation. In general, the cost of calculating secondary effects is expensive. In water simulation, foam, bubbles, and splash particles are typical [23], while fire simulation is a typical secondary effect. Recently, Kim et al. proposed the fire flake simulation technique for the first time in CG field [5]. This technique also calculates the motion of fire flake particles based on the underlaying simulation. This production pipeline is computationally expensive due to more than two simulations (for underlying simulation and secondary effects), and it is cumbersome to modify the underlying simulation in order to express the fire flake effects in accordance with the intention of the creator, and to recreate secondary effects after the simulation. Because our framework can quickly and easily express fire flake effects by receiving only generated fire videos, it can be applied to various industries such as game, VR/AR as well as VFX.

Although we did not apply the optimization method in the process of implementing the proposed algorithm, we can apply various video optimization algorithms to input video to improve the output : 1) By using a pyramid filter such as Gaussian pyramid, the resolution of the input image can be reduced and the computation can be processed quickly,

which can be utilized in the calculation of Fv^* and C_{pos} [24], [25]. 2) Similar to the pyramid method, the downsampling technique can be used to calculate Bv. Since it is important to extract the motion of fire from video, we do not have to worry about divergence-free or momentum-conservation conditions when calculating Bv. As a result, downsampling the grid resolution is not a problem for advecting fire flake particles. By optimizing the proposed method using the two methods mentioned above, we can express fire flake effects more quickly and efficiently.

VI. CONCLUSION

This paper proposes an image-based framework that can efficiently and realistically represent fire sprite animation. The force applied to the buoyancy field is modeled based on the feature vector of the image and its influence on the Y-axis direction, and the buoyancy vector field, which changes with time, is expressed by adding the buoyancy advection term. Based on this field, anisotropic fire-flake textures and motion blur were modeled to efficiently improve the visual detail of sprite animation.

Since there is no depth information in the input data of the proposed method, the interaction between the fire-flake particles and the object appears awkward. In addition, since it allows only videos and images as input data, it is difficult to perform scene editing such as changing viewpoint in scene or adding external forces. In the future, we plan to study how to improve the visual detail of fire-flakes effects using depth information and enable scene editing.

REFERENCES

[1] R. Fernando, E. Haines, and T. Sweeney, "GPU gems: Programming techniques, tips and tricks for real-time graphics," *Dimensions*, vol. 7, no. 4, p. 816, 2001.
[2] D. Kim, O.-Y. Song, and H.-S. Ko, "A practical simulation of dispersed bubble flow," *ACM Trans. Graph.*, vol. 29, no. 4, Jul. 2010, Art. no. 70.
[3] T. Kim, J. Lee, and C.-H. Kim, "Physics-inspired controllable flame animation," *Vis. Comput.*, vol. 32, pp. 871–880, Jun. 2016.
[4] J.-M. Hong, T. Shinar, and R. Fedkiw, "Wrinkled flames and cellular patterns," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007, Art. no. 47.
[5] T. Kim, E. Hong, J. Im, D. Yang, Y. Kim, and C.-H. Kim, "Visual simulation of fire-flakes synchronized with flame," *Vis. Comput.*, vol. 33, pp. 6–8, Jun. 2017.
[6] J. Gagnon, F. Dagenais, and E. Paquette, "Dynamic lapped texture for fluid simulations," *Vis. Comput.*, vol. 32, pp. 6–8, Jun. 2016.
[7] D. Enright, F. Losasso, and R. Fedkiw, "A fast and accurate semi-Lagrangian particle level set method," *Comput. Struct.*, vol. 83, pp. 6–7, Feb. 2005.
[8] C. Horvath and W. Geiger, "Directable, high-resolution simulation of fire on the GPU," *ACM Trans. Graph.*, vol. 28, no. 3, Aug. 2009, Art. no. 41.
[9] M. Cammarano and H. W. Jensen, "Time dependent photon mapping," in *Proc. 13th Eurograph. Workshop Rendering*, Jun. 2002, pp. 135–144.
[10] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen, "Physically based modeling and animation of fire," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 721–728, Jul. 2002.
[11] B. Solenthaler and R. Pajarola, "Predictive-corrective incompressible SPH," *ACM Trans. Graph.*, vol. 28, no. 3, Aug. 2009, Art. no. 40.
[12] M. Becker and M. Teschner, "Weakly compressible SPH for free surface flows," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, Aug. 2007, pp. 209–217.
[13] A. R. Fuller, H. Krishnan, K. Mahrous, B. Hamann, and K. I. Joy, "Real-time procedural volumetric fire," in *Proc. Symp. Interact. 3D Graph. Games*, May 2007, pp. 175–180.
[14] J. Stam, "Stable fluids," in *Proc. 26th Annu. Conf. Comput. Graph. Interact. Techn.*, Jul. 1999, pp. 121–128.
[15] N. Foster and R. Fedkiw, "Practical animation of liquids," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn.*, Aug. 2001, pp. 23–30.
[16] M. Müller, D. Charypar, and M. Gross, "Particle-based fluid simulation for interactive applications," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, Jul. 2003, pp. 154–159.
[17] M. Macklin and M. Müller, "Practical animation of liquids," *ACM Trans. Graph.*, vol. 32, pp. 104:1–104:12, 2013.
[18] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *J. Vis. Commun. Image Represent.*, vol. 18, pp. 109–118, Apr. 2007.
[19] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating eulerian fluid simulation with convolutional networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 3424–3433.
[20] M. Chu and N. Thuerey, "Data-driven synthesis of smoke flows with CNN-based feature descriptors," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017, Art. no. 69.
[21] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," Jun. 2018, *arXiv:1806.02071*. [Online]. Available: https://arxiv.org/abs/1806.02071
[22] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow," Jan. 2018, *arXiv:1801.09710*. [Online]. Available: https://arxiv.org/abs/1801.09710
[23] J.-H. Kim, J. Lee, S. Cha, and C.-H. Kim, "Efficient representation of detailed foam waves by incorporating projective space," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 9, pp. 2056–2068, Sep. 2017.
[24] P. J. Burt, "Fast filter transform for image processing," *Comput. Graph. Image Process.*, vol. 16, no. 1, pp. 20–51, 1981.
[25] J. L. Crowley, "A representation for visual information," Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-82-07, 1981.

JONG-HYUN KIM received the B.A. degree from the Department of Digital Contents, Sejong University, in 2008, and the M.S. and Ph.D. degrees from the Department of Computer Science and Engineering, Korea University, in 2010 and 2016, respectively.

He is currently an Assistant Professor with the Department of Software Application, Kangnam University. His current research interests include fluid animation and virtual reality.

JUNG LEE is currently an Assistant Professor with the Department of Convergence Software, Hallym University. His current research interests include augmented/virtual reality, fluid animation, and computer graphics.