Feedback Control of Fire Simulation based on Computational Fluid Dynamics

Syuhei Sato
UEI Research
(DWANGO Co., ltd.)

Keisuke Mizutani
Hokkaido University

Yoshinori Dobashi
Hokkaido University
UEI Research

Tomoyuki Nishita
UEI Research
Hiroshima Shudo University

Tsuyoshi Yamamoto
Hokkaido University

Abstract

Visual simulation of fire plays an important role in many applications, such as movies and computer games. In these applications, artists are often requested to synthesize realistic fire with a particular behavior. To meet such requirement, we present a feedback control method for fire simulations. The user can design the shape of fire by placing a set of control points. Our method generates a force field and automatically adjusts a temperature at a fire source, based on user specified control points. Experimental results show that our method can control the fire shape.

Keywords: fire simulation, grid-based simulation, PI-control, editing shape and motion

1 Introduction

In computer graphics, many methods have been proposed for simulating natural phenomena such as smoke, fire, water, and so on [1, 2, 3, 4, 5]. Realistic fluid animations can be created by solving Navier-Stokes equations. Therefore, in recent years, fluid simulation has become an indispensable tool for creating scenes containing fluids in entertainment applications, such as movies and TV games. In these applications, animators often want to generate their desired motions while preserving realistic visual appearances of fluids. One effective approach to achieve this is to control the motions of the fluids based on numerical fluid analysis. Several methods have therefore been proposed for controlling the fluid simulation [6, 7, 8]. In these previous methods, the motions of smoke or water are controlled into user-specified shapes using external forces.

Visual simulation of fire is important in creating synthetic images of a scene including, for example, a burning house and a dragon breathing out fire. Computational fluid dynamics (CFD) is effective for synthesizing realistic fire as well [4, 9, 10]. Although methods using CFD can generate realistic fire, there are many physical parameters that influence the motions of the simulated fire. Therefore, animators have to repeatedly execute the fluid simulation with different parameter settings until the desired fire motions are obtained. Moreover, the shapes and motions of the simulated fire that can be created by adjusting the simulation parameters are limited and, in most cases, it is almost impossible to create the desired animation of fire.

Several methods have been proposed for procedurally creating desired motion of fire [11, 12]. These procedural methods allow us to create desired shapes of the fire at low costs. However, these methods are not based on the underlying physics governing the fire motion. Therefore, the resulting animation is less realistic than that created using physically-based simulation.

Our purpose in this paper is to control CFD-

based fire simulation in order to create realistic fire animations with desired shapes and motions. In many commercial products, such as Autodesk Maya and 3ds MAX, external forces are generally used for controlling fluid shapes and motions. The external forces in these software are often defined by using simple 3D primitives, such as cylinder and sphere; the forces are generated inside these primitives. However, in those products, magnitudes of the external forces must be adjusted manually by the user, which is not an easy task. A straightforward solution to this problem is to use the previous methods for controlling smoke or water. However, we found that those previous methods could not control fire simulation faithfully since the complex motion of fire significantly depends on temperature distribution. The previous methods for smoke and water did not take into account temperature in their control methods because it is not important very much in simulating their motions.

We propose a new method that addresses the above problem by automatically controlling the external forces as well as the temperature of fire source. Our method mainly controls fire simulation by generating external force fields. The user can design the shapes and motions of fire as if the user drags the flame region of the fire by drawing a set of line segments. Then, the external force fields are generated around the line segments and their magnitudes are automatically adjusted so that the flame region reaches the positions of the end points of the line segments. However, we sometimes observed that this method generated extremely strong forces to make the flame region forcibly reach the target position, resulting in unnatural motions. In order to solve this problem, our method additionally controls the temperature of fire source to adjust the overall size of the flame region. This makes it possible to control the fire shapes/motions with the minimal external forces. With these two control mechanisms, the user can design the shape and motion of fire while maintaining its realistic appearance.

Our main contributions include:

1. controlling CFD-based fire simulation to create animations with desired shapes and motions,
2. application of the feedback control mechanism to fire simulation, and
3. controlling temperature of fire source, in addition to external forces, to enhance the realism of the controlled simulation.

One of the limitations of our method is that our method would take long time to reach the converged state when we use inappropriate control parameters for the feedback control mechanism. In this case, interactive editing of fire simulation to create desired animations would become difficult. Although we provide appropriate parameter settings used for our experiments in this paper, the user may need to spend time on adjusting the control parameters.

Several methods have been proposed for simulating realistic fire but we employ a relatively simple method. We simulate fire as a heated gas generated by combustion. Although the behavior of the simulated fire is slightly less realistic than that by a more physically-based method, such as the two-fluid model [4], our method is far more efficient and is easy to implement. We employ the simple method because our primary purpose is to synthesize the desired shape with plausible appearances. Our method is fully implemented on the GPU and the user can edit the fire animation interactively.

2 Related Work

There are many methods for visual simulation of fire. They are roughly classified into two categories: the procedural approach and physically-based approach.

Methods based on the procedural approach have focused on efficiency and controllability. Particle systems are the most widely used method. The earliest reported fire model, presented by Reeves [13], used a particle system to animate fire. This method required a large number of particles to achieve natural visual effects. After this work, some methods have been proposed for procedural modeling of fire animation [11, 12, 14]. Although the computational cost for the procedural methods is low and the desired fire shapes are easy to generate, some important visual features, such as vortex motions and detailed turbulence motions, cannot be reproduced.

On the other hand, methods based on the physically-based approach can generate realistic visual results with evolving and dynamic fire. Various methods for simulating fire have been developed based on incompressible fluid solvers [4, 9, 10]. Although these methods have the potential to generate realistic fire, many physical parameters have to be adjusted to obtain the desired results. Adjusting the parameters manually to synthesize fire with desired shapes and motions is time-consuming or, furthermore, almost impossible.

Some researchers have developed methods for controlling simulation of phenomena related to fluids. Treuille et al. [15] developed a method for keyframe control of smoke simulation. McNamara et al. [7] proposed a method for controlling smoke and water by using an adjoint method. Hong et al. [6] proposed to use a geometric potential field generated from given target shapes to generate control forces. Fattal and Lischinski [16] developed a target-driven method to control smoke simulation by using some external forces computed from a potential field. Shi and Yu proposed a method using a feedback force field and the negative gradient field of geometric potential function for creating the animation of water towards to the rapidly changing targets [8]. Shi and Yu also developed a method for controlling a shape of a smoke, such that a smoke surface is matched a target surface, using velocity constraints [17]. Thürey et al. [18] introduced control particles for controlling the motion of water. Kim et al. [19] proposed a method to advect smoke along a user-specified path. Dobashi et al. [20] presented a feedback control method for a cumuliform cloud simulation. Raveendran et al. [21] proposed a liquid control method using control meshes. These control methods, however, do not cover fire simulation. Controllable fire has been an active research area in the last few years. Lever and Komura [22] controlled fire simulation with textured force for generating volumetric patterns similar to input textures on the fire volume. Hong et al. [23] introduced a method for controlling blue core for modeling fire propagation along complex curves or surface. Zhang et al. [24] applied high geometric motion constraints to the fire animation. Bangalore and House [25] proposed artistic control of flames using a set of curves drawn by the user. This method deals with target shapes formed by a set of curves only. Kim et al. [26] focused on controlling fire shapes into user-specified 3D objects. However, in these fire control methods, there are no methods for controlling the motion of fire based on computational fluid dynamics.

3 Fire Simulation

Before describing our method, this section explains the method for numerical simulation of fire used in our method. We simulate fire by using a grid-based approach. The simulation space is subdivided into a grid, and a fire source is placed at an arbitrary position in the simulation space. Velocity $\mathbf{u}$ and temperature T are assigned at each grid point. Motions of the fire are simulated by the following way. Velocity $\mathbf{u}$ is updated at each time step by solving the following Navier-Stokes equations.

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p + \mathbf{f}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where t is time, p is fluid pressure, and $\mathbf{f}$ represents any external forces such as gravity, wind, and buoyancy forces. For the fire simulation, temperature and buoyancy forces are calculated by using the previous techniques proposed by Nguyen et al. [4]. Temperature T is calculated by the following equation.

$$\frac{\partial T}{\partial t} = -(\mathbf{u} \cdot \nabla)T - c_r(\frac{T - T_{amb}}{T_{max} - T_{amb}})^4 + T_{src}, \tag{3}$$

where T_{amb} is ambient temperature, T_{max} is the maximum temperature, c_r is a cooling constant, and T_{src} is a temperature added at the source of the fire. As an external force, the buoyancy force is taken into account. The buoyancy force $\mathbf{f}_{buo}$ is given by:

$$\mathbf{f}_{buo} = \kappa_b(T - T_{amb})\mathbf{y}, \tag{4}$$

where κ_b is the coefficient for the buoyancy and $\mathbf{y}$ is a unit vector pointing upward vertical direction. Velocities at the grid points around the fire source are fixed on that of the fire source.

By solving the above equations numerically, the temperature field is obtained at each time

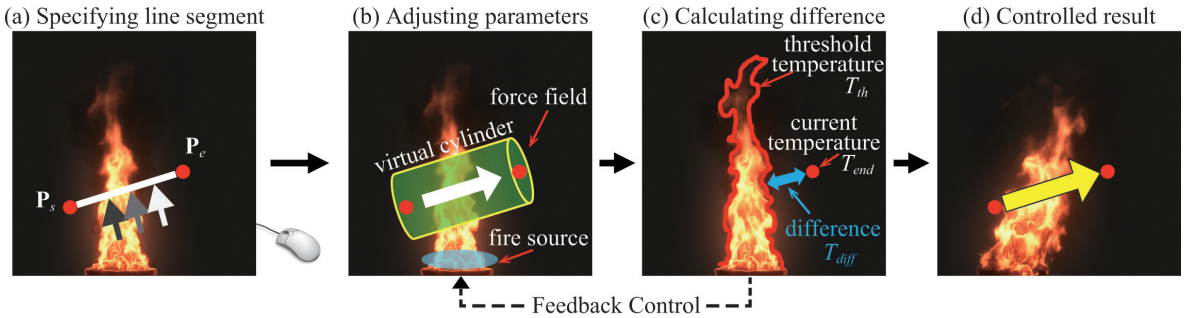


Figure 1: Overview of our method.

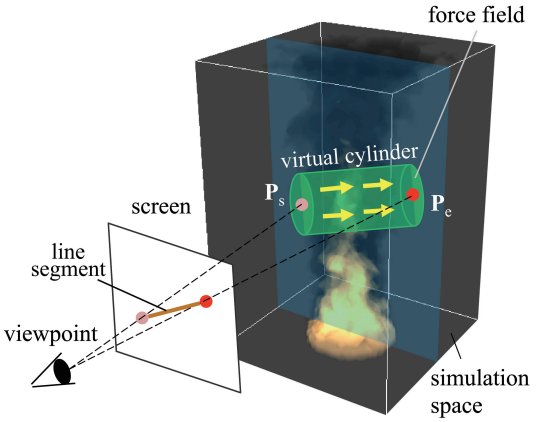


Figure 2: Our fire control system.

step. A user-specified threshold temperature, T_{th} is used to define the boundary between the flame and the smoke regions. That is, a point is inside the flame region if temperature at the point is greater than the threshold temperature T_{th}.

4 Our Method

4.1 Overview

Fig. 1 illustrates an overview of our method. The user can interactively design the shape and the motion of synthetic fire as if he or she drags the flame region by drawing a line segment on the screen (see Fig. 1(a)). Our method controls the simulation so that the flame region reaches to the end point of the line segment in the following way. When the user draws the line segment, a virtual cylinder is automatically created around the line segment (see Fig. 1(b)), inside which a force field is generated to control the simulation. Our method automatically adjusts the magnitude of the force field and the temperature of the fire source by a feedback control mechanism. Our feedback controller tries to minimize the distance between the flame region and the end point of the line segment. This is equivalent to minimizing the difference between temperature T_{end} at the end point and the threshold temperature T_{th} defining the boundary of the flame region (Fig. 1(c)). Through our control mechanism, the fire shape is deformed so that the boundary of the flame region reaches to the end point of the line segment (Fig. 1(d)). Note that the user can draw multiple line segments at arbitrary positions to design the fire shape.

The virtual cylinder and the corresponding force field are generated in the following way. The two end points of the line segment drawn by the user are projected onto the plane that is placed at the center of the simulation space and is perpendicular to the view direction (see Fig. 2). Let us denote these projected points for i-th line segment by $\mathbf{P}_s(i)$ and $\mathbf{P}_e(i)$, respectively. The virtual cylinder is created with a predetermined radius r. The top and bottom centers of the cylinder are $\mathbf{P}_s(i)$ and $\mathbf{P}_e(i)$, respectively (Fig. 2). The radius of the cylinder is experimentally determined. In this paper, we set the radius to 20 times grid interval used for the simulation. We need further investigation on the choice of the radius but we have not encountered any significant problems with the above setting. The force $\mathbf{f}_e(i)$ inside the virtual cylinder is computed by the following equation.

$$\mathbf{f}_e(i) = c_f(i)(\mathbf{P}_e(i) - \mathbf{P}_s(i)), \qquad (5)$$

where $c_f(i)$ is a coefficient for adjusting the

magnitude of i-th force field. Note that $\mathbf{f}_e(i)$ is zero outside the cylinder. Our method controls the coefficient $c_f(i)$ as well as the temperature of the fire source, T_{src}, so that temperature at $\mathbf{P}_e(i)$ becomes T_{th}.

4.2 Feedback Control

We employ a proportional-integral-derivative controller (PID controller) to automatically adjust the magnitude of the force field and the source temperature. The PID controller is one of the most popular feedback control methods. It evaluates the error (or difference) between the current state and the target state, and tries to minimize the error by continuously adjusting the control variables over time. The PID controller consists of three components: P-controller, I-controller, and D-controller. The P-controller updates the control variables in proportion to the error. The I- and D-controllers modify the control variables in proportion to the accumulated error and the derivative of the error, respectively. In our case, the control variables are c_f and T_{src}, and the error is computed as the difference between temperature T_{end} at $\mathbf{P}_e$ and the threshold temperature T_{th}. We found that the D-controller does not contribute very much, since the motion of the fire changes randomly and the derivative of the error does not make any senses. In the following, we first explain the control of the magnitude of the force field and then describe the control of fire source temperature.

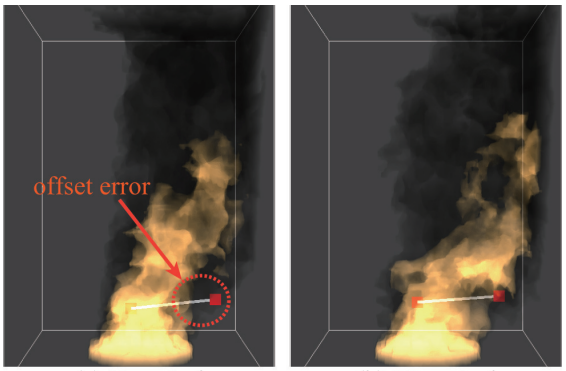
As we mentioned in the previous subsection, a force field is generated inside the virtual cylinder for each line segment. We assign a PI-controller for each force field and the controllers work independently with each other. Let us explain our control mechanism for i-th force field. The magnitude of the force field is controlled by the PI-controller which updates $c_f(i)$ at each time step according to the following equations.

$$c_f(i) = K_P T_{diff}(i) + K_I \int_0^{t_d} T_{diff}(i) dt_d, \tag{6}$$

where

$$T_{diff}(i) = T_{th} - T_{end}(i), \tag{7}$$

and $T_{end}(i)$ is the temperature at $\mathbf{P}_e(i)$. The first term on the right is the P-controller that updates



(a) P control (b) PI control

Figure 3: Comparison of results controlled by P-controller and PI-controller for force field control.

$c_f(i)$ in proportion to the temperature difference. K_P is called a proportional gain. The second term, the I-controller, updates $c_f(i)$ in proportion to the accumulated difference over time and gives the accumulated offset that should have been corrected previously. The I-controller contributes when the accumulated difference becomes large. t_d is the duration for the accumulation and K_I is called the integral gain. These controllers work in the following way. When the boundary of the flame region becomes close to $\mathbf{P}_e(i)$, the temperature difference $T_{diff}(i)$ becomes very small. In this case, the P-controller cannot contribute to the updating of $c_f(i)$. As a result, when using the proportional controller only, small gaps between the boundary of the flame region and $\mathbf{P}_e$ are left. The second term removes the gaps and updates c_f until the difference becomes zero. The control parameters K_P and K_I need to be specified by the user. In this paper, we determine these parameters experimentally. Fig. 3 shows the effect of these controllers. Fig. 3(a) shows the result using only the proportional controller. The flame region cannot reach $\mathbf{P}_e(i)$. This problem is resolved by using the I-controller as shown in Fig. 3(b).

The above method for controlling the force field sometimes produces unnaturally strong forces. This happens particularly when the user places $\mathbf{P}_e(i)$ at a position far from the boundary of the flame region. This problem is resolved by changing the temperature of the fire source. For example, increasing source temper-

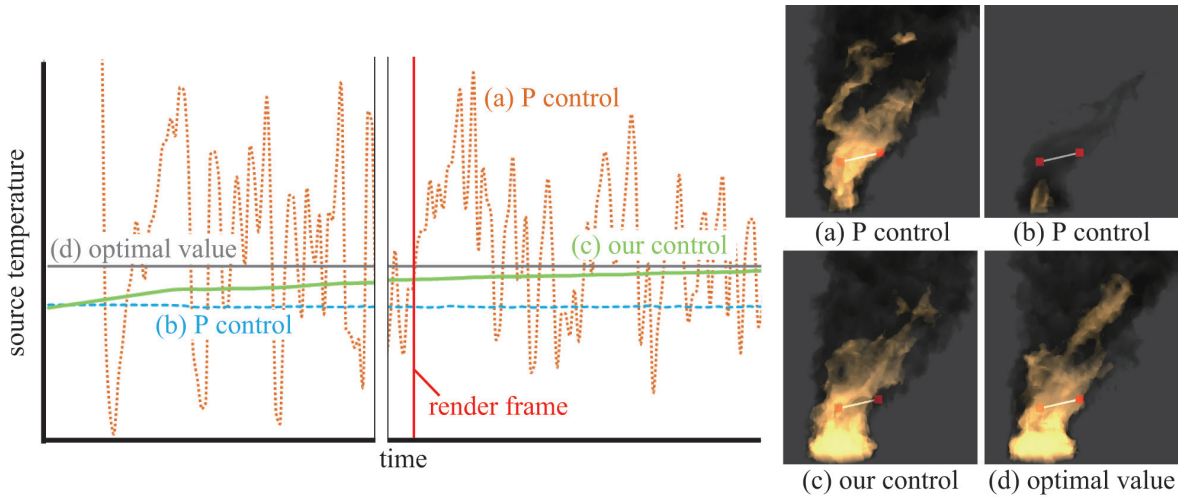


Figure 4: Comparison of results controlled by P-controller and our control (I-controller) for source temperature control.

ature results in an increase in the overall temperature of the fire and, as a result, the size of the flame region increases as well. This reduces the distance between the flame region and the user-specified target position, or the end point of the user-specified line segment. Therefore, we do not need to use strong forces to make the flame region reach the target position. Based on this idea, our method controls the source temperature and adjusts the overall size of the fire automatically. The source temperature is updated to minimize the average of the temperature differences at all end points of the line segments. We use an I-controller with an initial offset temperature, that is,

$$T_{src}(i) = T_0 + c_I \int_0^{t_d} \bar{T}_{diff} dt_d, \qquad (8)$$

where

$$\bar{T}_{diff} = \frac{1}{n} \sum_i T_{diff}(i), \qquad (9)$$

and T_0 is the initial offset temperature and c_I is an integral gain for the source temperature. Although the P-controller is the simplest and the most popular controller, we do not use it. We found that the P-controller was not suitable for our purpose. When using the P-controller, the source temperature is set to be proportional to the average of the differences at the end points of the user-specified line segments. Therefore, the source temperature becomes zero when the

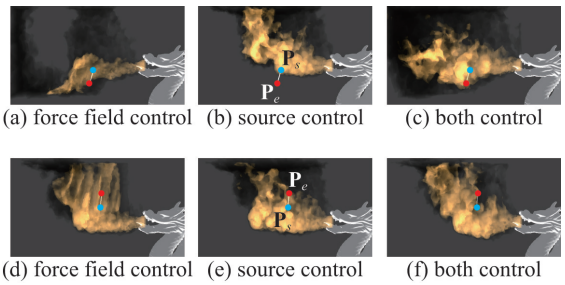


Figure 5: Comparisons of results created by using force field controller only, source temperature controller only, and both controllers.

average difference is zero, that is, the fire extinguishes. Then the flame region starts to shrink and the source temperature increases again. This causes an oscillation of the source temperature, resulting in an unnatural animation. We can avoid this problem by using the I-controller that makes the source temperature gradually converge to an optimal state. The initial offset temperature T_0 is used to ensure that fire exists even when there are no user-specified line segments.

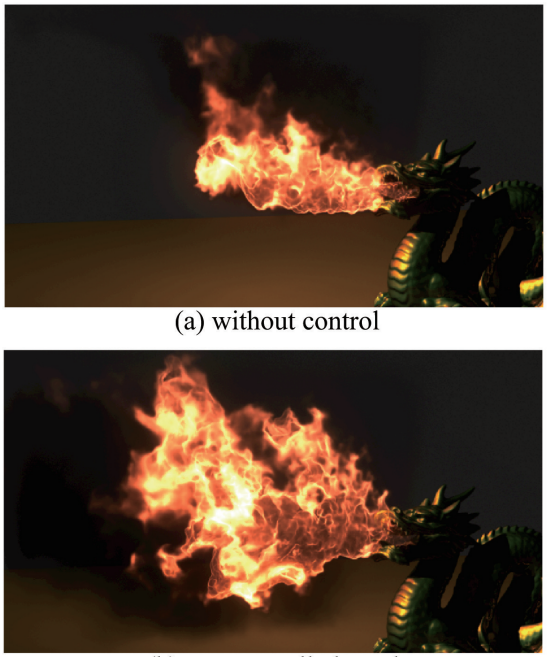
Fig. 4 shows comparisons of fire simulation controlled by P- and I-controllers. We draw a single line segment as shown in the figure. The magnitude of the external force is fixed and is not controlled. Figs. 4(a) and (b) show results obtained by using the P-controller only, and Fig. 4(c) is generated by using our I-controller.

Fig. 4(d) is a reference solution that was created without controlling the source temperature; the source temperature is fixed at $T_{src} = 1.0$ that was determined experimentally so that the flame region reaches the end point as shown in the figure. We call this temperature ($T_{src} = 1.0$) an optimal temperature. When using the P-controller (Figs. 4(a) and (b)), the source temperature is updated by: $T_{src} = T_0 + c_P \bar{T}_{diff}$, where c_P is a proportional gain. T_0 is set to 1.0 and 0.1, and the proportional gain is set to 1.0 and 0.01, respectively, for Figs. 4(a) and (b). The graph on the left shows three plots representing temporal transitions of the source temperature for Figs. (a) through (c). The P-controller failed to control the source temperature adequately: the source temperature oscillates (Fig. 4(a)) or is too small (Fig. 4(b)). In contrast, our I-controller (Fig. 4(c)) successfully controls the source temperature; the temperature is gradually increased and is converged to the optimal temperature. These results clearly demonstrate that the I-controller is suitable for the source temperature control.

5 Result

This section shows some examples created by our method. We used a desktop PC with Intel Core i7 (8GB of RAM) and NVIDIA GeForce GTX 780 as a GPU to compute all the examples shown in this section. Please refer to the accompanying video file for animations of the examples shown in this section.

Fig. 5 shows results created by our method under different situations. Figs. 5(a) and (d) are created by using the force field control only. Figs. 5(b) and (e) are created by controlling the source temperature only. Figs. 5(c) and (f) are created by using both controllers. In Figs. 5(a), (b) and (c), we drew the line segment such that the fire shape expands to lower direction. In contrast, in Figs. 5(d), (e) and (f), the line segment was drawn such that the fire shape expands to upper direction. In Fig. 5(a), the magnitude of the force field is too large, so the motion of the fire looks unnatural. In Fig. 5(b), the flame region does not reach the position of the end point of the line segment. By controlling the force field and the source temperature, the flame region successfully reaches the end point and the fire motion looks very natural. Next, in Fig. 5(d), the magnitude of the force field becomes too large to make the flame region forcibly reach to the end point of the line segment. This problem is caused since the source temperature is too small. By controlling the source temperature, the flame region naturally reaches to the end point, as shown in Figs. 5(e) and (f). In this case, controlling only the source temperature may be sufficient (Fig. 5(e)) but controlling both of the force field and the source temperature can also produce the natural animation (Fig. 5(f)). The user can choose his/her preferable result.



(a) without control

(b) our controlled result

Figure 6: Fire from dragon.

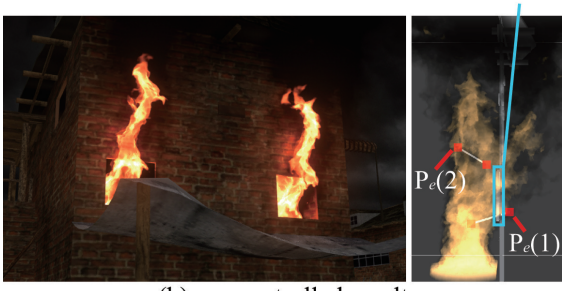
Figs. 6, 7 and 8 show some practical examples of fire simulation controlled by our method. The simulation space is subdivided into 128x192x128 (Figs. 6 and 7) or 128x192x192 (Fig. 8) grid points. The computation time for each step took about 0.1 seconds. Figs. 6(a), 7(a) and 8(a) are fire simulations without any control, and Figs. 6(b), 7(b) and 8(b) are results created by using our control method. In Fig. 6, the shape of fire from the Dragon is edited so that the shape becomes wider. Fig. 7 shows an example of a burning house. By using our

(a) without control

(b) our controlled result

Figure 7: Burning house.

(a) without control



(b) our controlled result

Figure 8: Fire spout from windows of burning house.

method, we can make fire wrapping around the house like Fig. 7(b) without computing interactions between fire and the house. In Fig. 8, we try to control motions of fires such that the fires spout from windows of the burning house, without computing the interactions. The insets on the right in the figure show the line segments we specified to create this animation. By using our method, plausible fire animations with desired shapes and motions can be created efficiently.

6 Conclusions

We have proposed a method to control fire simulation based on computational fluid dynamics. Our method generates the external force fields and automatically adjusts the temperature at the source of the fire. The user can design the shape and motion of fire by drawing a set of line segments. In order to generate the desired flow and fire shape, the external force fields and the source temperature are controlled according to the user-specified line segments. Our method provides a simple way to generate realistic fires with desired shapes and motions.

In our future work, we plan to extend our method to other types of fluid simulations such as particle-based methods.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP15H05924.

References

[1] R. Bridson. *Fluid Simulation for Computer Graphics*. AK Peters, 2008.

[2] B. E. Feldman, J. F. O'Brien, and O. Arikan. Animating suspended particle explosions. In *Proceedings of ACM SIGGRAPH 2003*, pages 708–715, 2003.

[3] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, pages 23–30, 2001.

[4] D. Q. Nguyen, R. Fedkiw, and H. W. Jensen. Physically based modeling and animation of fire. *ACM Transactions on Graphics*, 21(3):721–728, 2002.

[5] Jos Stam. Stable fluids. In *Proceedings*

of ACM SIGGRAPH 1999, Annual Conference Series, pages 121–128, 1999.

[6] J. Hong and C. Kim. Controlling fluid animation with geometric potential. *Computer Animation and Virtual Worlds*, 15(3-4):147–157, 2004.

[7] A. McNamara, A. Treuille, Z. Popovic, and J. Stam. Fluid control using the adjoint method. *ACM Transactions on Graphics*, 23(3):449–456, 2004.

[8] L. Shi and Y. Yu. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 229–236, 2005.

[9] Jeong-Mo Hong, Tamar Shinar, and Ronald Fedkiw. Wrinkled flames and cellular patterns. *ACM Transactions on Graphics*, 26(3):Article 47, 2007.

[10] C. Horvath and W. Geiger. Directable, high-resolution simulation of fire on the gpu. *ACM Transactions on Graphics*, 28(3):Article 41, 2009.

[11] A. R. Fuller, H. Krishnan, K. Mahrous, B. Hamann, and K. I. Joy. Real-time procedural volumetric fire. In *Proceeding of the 2007 symposium on Interactive 3D graphics and games*, pages 175–180, 2007.

[12] A. Lamorlette and N. Foster. Structural modeling of flames for a production environment. *ACM Transactions on Graphics*, 21(3):729–735, 2002.

[13] W. T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, 1983.

[14] P. Beaudoin, S. Paquet, and P. Poulin. Realistic and controllable fire simulation. In *Proceedings of Graphics Interface 2001*, pages 159–166, 2001.

[15] A. Treuille, A. McNamara, Z. Popovic, and J. Stam. Keyframe control of smoke simulations. *ACM Transactions on Graphics*, 22(3):716–723, 2003.

[16] R. Fattal and D. Lischinski. Target-driven smoke animation. *ACM Transactions on Graphics*, 23(3):439–446, 2004.

[17] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Transactions on Graphics*, 24(1):140–164, 2005.

[18] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 7–12, 2006.

[19] Y. Kim, R. Machiraju, and D. Thompson. Path-based control of smoke simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 33–42, 2006.

[20] Y. Dobashi, K. Kusumoto, T. Nishita, and T. Yamamoto. Feedback control of cumuliform cloud formation based on computational fluid dynamics. *ACM Transactions on Graphics*, 27(3):Article 94, 2008.

[21] K. Raveendran, N. Thuerey, C. Wojtan, and G. Turk. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 255–264, 2012.

[22] J. Lever and T. Komura. Real-time controllable fire using textured forces. *The Visual Computer*, 28(6):691–700, 2012.

[23] Y. Hong, D. Zhu, X. Qiu, and Z. Wang. Geometry-based control of fire simulation. *The Visual Computer*, 26(9):1217–1228, 2010.

[24] Y. Zhang, C.D. Correa, and K.-L. Ma. Graph-based fire synthesis. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 187–194, 2011.

[25] A. Bangalore and D.H. House. A technique for art direction of physically based fire simulation. In *Proceedings of the*

Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging, pages 45–54, 2012.

[26] Taehyeong Kim, Jung Lee, and Chang-Hun Kim. Physics-inspired controllable flame animation. *The Visual Computer*, 32(6–8):871–880, 2016.