Still-Frame Simulation for Fire Effects of Images

Minjung Son¹  Byungmoon Kim²  Gregg Wilensky²  Seungyong Lee¹
¹POSTECH  ²Adobe

Abstract
We propose various simulation strategies to generate single-frame fire effects for images, as opposed to multi-frame fire effects for animations. To accelerate 3D simulation and to provide a user with early hints on the final effect, we propose a 2D-guided 3D simulation approach, which runs a faster 2D simulation first, and then guides 3D simulation using the 2D simulation result. To achieve this, we explore various boundary conditions and develop a constrained projection method. Since only the final frame will be used while intermediate frames are abandoned, earlier intermediate frames can take larger time steps and have large noise applied, quickly generating turbulent flow structures. As the final frame approaches, we increase the flow quality by reducing the time step and not adding any noise. This adaptive time stepping allows us to use more computational resource near or at the final frame. We also develop divergence and buoyancy modification methods to guide flames along arbitrary, even physically implausible, directions. Our simulation methods can effectively and efficiently generate a variety of fire effects useful for image decoration.

Categories and Subject Descriptors (according to ACM CCS): I.6.8 [Simulation and Modeling]: Types of Simulation—Visual; I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Introduction
Fire effects are found as still-frame effects in a variety of printed materials, such as books, magazine and newspaper commercials, and posters, as well as multi-frame effects in animations. Artists and designers use still-frame fire effects to provide impressive images with fire’s fascinating and distinguishing color and style. They insert fire as an object or background into an image using a wide range of fire styles. They also make fire burn following specific shapes for emphasizing or decorating image borders, object boundaries, title letters, logos, etc.

Typically, still frame fire effects have been generated manually. The most common method is to copy-and-paste a source fire image, typically a real image, that has desired details [Ch-11]. Artists overlay fire onto the image, and manually perform cut, paste, delete, rotate, and scale operations in order to obtain desired shape, flame direction and details. This method generates impressive results, but does not generate natural flames with realistic fuel burns, heat rises, and turbulent structures with various scales. Artists working in photo-realistic styles [Mei89,MD93] draw paintings that appear like photographs. Such artists can depict photo-realistic fire, but it requires much more specialized skills and expensive efforts.

Still frame fire effects may be automatically generated by procedural methods [Fer04, FKM⁺07]. Using a procedural approach, however, it is difficult to generate fire flames with natural details, such as laminar to turbulent flow transitions, volumetric vorticities, and naturally mixed large, medium and small-scale details.

A simulation-based method can be used for realistic and highly detailed still-frame fire effects. Fire can be simulated by spreading fuel to follow a target shape in the image, burning the fuel, and then guiding the flame to a desired direction. These operations enable an intuitive control over the fire shape and the flow of fire details.

In any effect simulation, user interaction with instant feedback is beneficial. For fire image effects, interactivity allows the user to control the flame direction and style during the simulation and to instantly choose the final frame. An interactive fire simulation system could be developed with low-resolution or 2D simulation, but the resulting quality would not be satisfactory, due to lack of volumetric details. High-resolution 3D simulation produces high quality detailed fire, but it would be too slow for an interactive system.
This quality and speed trade-off is a critical factor in utilizing the simulation for image effects. To fulfill these two essential but conflicting properties, we propose a new simulation strategy leveraging the fact that our target is just one scene and that fire effects synthesized for images have relatively shallow depths. The idea is 2D-guided 3D simulation, where we first carry out a fast 2D simulation to provide a user with interactivity, then run a more expensive 3D simulation guided by the 2D simulation result to generate a volumetric and detailed final frame. In addition, we develop acceleration techniques for our simulation, such as adaptive time stepping and noisy initialization.

Traditional simulation methods have limitations on being directly used for fire image effects, because they aimed to generate mostly physically valid animations. As artistic effects, the final results of our fire simulation may need to be guided into physically implausible configurations. Among various possibilities, for example, the user may want to obtain the fire blazing toward a point from all directions, as the focus is on the artistic purpose rather than physical correctness. To help this, we propose modifications of the simulator to handle non-physical flows by adding adaptive divergence or force.

In summary, we present an effective and efficient still-frame fire simulation method that is useful for producing artistic fire effects with the following contributions:

- We propose a novel approach for fire simulation tailored to create still-frame image effects, instead of full-frame animations.
- We develop a novel simulation strategy, 2D-guided 3D simulation, fulfilling both interactivity and quality needed for image effect generation.
- We present acceleration techniques for still-frame simulation: adaptive time stepping and noisy initialization.
- We improve the simulation to handle physically implausible but artistically desirable cases of various fire effects.

2. Related Work

A few software applications are available to help users add fire effects into images. However, they are usually based on procedural approaches [Fer04, FKM’07, HZQW10]: a user adds fire blades with specified widths and heights, resulting in less realistic fire details. It is difficult to add fire flames with natural behaviors, such as realistic transition of laminar to turbulent flow and volumetric details with vorticities.

Simulation has been used to produce a variety of realistic and artistic effects for images. To name a few, physically-based or observational models were proposed for simulating watercolor [CAS’97], pencil drawing [SB00], oil painting [CBWG10], and ink dispersion [CT05]. However, to the best of our knowledge, using physically-based fire simulation for image effects was not studied previously. On the other hand, fire simulation has been broadly used for animation with evolving and dynamic fires. We briefly review fire simulation methods here.

Fire simulation is based on either a compressible or an incompressible flow solver. In general, compressible solvers are difficult to use since they require very small time steps to be stable; incompressible solvers do not suffer from this limitation [Sta99]. When compressible flow effects such as shock do not exist at all, i.e., the fire burns slowly without explosion, incompressible fluid solvers become a much more attractive option thanks to stability and efficiency. Therefore, various fire simulation methods have been developed based on incompressible fluid solvers [MBR’01, TOT’03, FOA03, HG09]. For the same reason, we use an incompressible fluid model in this paper.

To make heated gas rise, almost all fire and smoke simulation methods use buoyancy force [FSJ01, SRF05, NFJ02]. This is because applying buoyancy is simple, buoyancy direction is directly controllable, and the output is of high quality. Two computational models are used for fuel burning. With the thin-flame model [Mar64, NFJ02, HSJ07], liquid or solid fuel is gasified to a gas fuel that burns and expands only at a thin region, which is implemented as a jump condition. Alternatively, gas fuel can be advected along the velocity field and burnt if the temperature and/or oxygen mixture are high enough [MK02, HG09]. Compared to the thin flame model, this model produces more smoke-like fire behaviors, i.e., vorticities - an incompressibility artifact, particularly when incompressibility is strictly enforced using a conjugate gradient (CG) solver. This model is used in this paper as it can be implemented more easily and faster due to the lack of the level set and jump condition.

Using 2D simulation to accelerate 3D simulation was studied before. Rasmussen et al. [RNGF03] used a set of 2D simulation slices, where volumetric flow is produced by adding structured noise and then exchanging momentum between slices using passive particles for all animated frames. Similarly, Horvath and Geiger [HG09] used multiple slices of 2D simulation grids with effective slice refinement. However, with the limited numbers of 2D simulation slices, these methods are not full 3D and the volumetric effects are not rich enough. In contrast, we run 2D simulation first, use it as a preview, and run full 3D simulation for a short period of time using the 2D result as a constraint applied to a single slice. As a result, our method can generate fully volumetric patterns of fires.

Regarding simulation control, there are several research investigations devoted mostly to smoke animations [TMPS03, FL04]. These methods generate animations that match specific frames with the input. Lever and Komura [LK12] gave control to fire simulation with textured force for generating volumetric patterns similar to input textures on the fire volume. Zhang et al. [ZCM11] suggested fire animation control with high-level geometric mo-
tion constraint. While these approaches target at controlling the progress of a fire animation, our method is tailored to produce a desired single frame for fire image effect.

3. Governing Equations

For Newtonian physics and mass conservation, we use the Navier-Stokes and continuity equation solver using operator splitting, semi-Lagrangian advection, an implicit viscosity solver, and pressure projection. This solves for the velocity $\mathbf{u}$ and pressure $P$ in the Navier-Stokes equation

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} + \nu \nabla^2 \mathbf{u} + \frac{\nabla P}{\rho} + \mathbf{f}, \quad (1)$$

where $\nu$ is the kinetic viscosity, $\rho$ is density, $\mathbf{f}$ is body force, and $t$ is time. Let $T$ be temperature and let $\mathcal{H}$ be internal energy. Assuming that changes in $\mathcal{H}$ are approximated with constant heat capacity $c_v$, and heat energy rate by the fuel burn rate $\mathcal{Q}$ as $\frac{\partial \mathcal{H}}{\partial t} = \rho c_v \frac{\partial T}{\partial t} + \mathcal{Q}$, energy transport in an infinitesimally small control volume yields the energy equation

$$\rho \frac{D \mathcal{H}}{Dt} = \mathbf{k} \nabla^2 T - P \nabla \cdot \mathbf{u} + \Phi \approx \rho c_v \frac{\partial T}{\partial t} + \mathcal{Q}, \quad (2)$$

where $D/ Dt$ is the material derivative, $P \nabla \cdot \mathbf{u}$ is the work done by pressure, and $\Phi$ is the temperature increase by velocity dissipation. Since the energy released during fuel burn $\mathcal{Q}$ is much greater than $P \nabla \cdot \mathbf{u} + \Phi$, we can ignore these terms.

To solve for density $\rho$, we would couple the above two and the continuity equations with the gas state equation, $\rho = \rho(P, T)$ (for ideal gas, $P = \rho RT$). However, variable density requires a compressible flow solver that is computationally inefficient [KGF10]. Therefore, we use a more practical model that ignores density variation first and then adds buoyancy in high temperature gas regions.

Let $H$ be the heating values of a fuel, i.e., energy released when a unit mass of the fuel is burnt. For example, for methane gas, $H = 50.1\text{MJ/kg}$, and for wood, $H = 20.0\text{MJ/kg}$. Let $F_b$ be the rate at which fuel is burnt. Then the heat energy is released at the rate $\mathcal{Q} = HF_b$.

In addition, we model the radiational cooling rate as $C_T T^4$, similarly to [NFJ02], with $C_T$ being a constant that a user can choose. This model cools high temperature regions very rapidly, but cools lower temperature regions very slowly. In fact, low temperature regions should be cooled by the temperature diffusion term $\mathbf{k} \nabla^2 T$ in Eq. (2). However, diffusion will smooth out temperature details - an effect that is undesirable in our application, where rich details of fire are targeted for image effects. Therefore, for low temperature cooling, we use the Newton’s Law of cooling model, which decays the temperature exponentially down to the ambient temperature ($T = 0$ in our experiments). This is equivalent to replacing $\mathbf{k} \nabla^2 T$ by an exponential decay term $-d_T T$, where $d_T$ is the user controllable decay coefficient. Putting all of these together, the heat equation Eq. (2) is modified with heating and cooling as

$$\frac{\partial T}{\partial t} = -\mathbf{u} \cdot \nabla T + \frac{H}{\rho c_v} F_b - C_T T^4 - d_T T. \quad (3)$$

Since gas fuels are advected along the velocity field, and diffused while being burnt at the rate $F_b$, the fuel behavior is modeled as

$$\frac{\partial F}{\partial t} = -F_b - \mathbf{u} \cdot \nabla F + \mu_F \nabla^2 F, \quad (4)$$

where $\mu_F$ is the fuel diffusion coefficient. If $T < T_{\text{ignition}}$, the burn rate $F_b$ is zero. In general, $F_b$ is a function of stochastic air/fuel mixture, oxygen density, and burn rate of the fuel. In this paper, we simply assume $F_b$ is a constant that the user chooses to decide how fast the fuel should burn.

4. 2D Simulation Modification

Based on the fire simulation model in Sec. 3, we solve Eq. (1) by the stable fluids method [Sta99]. With the traditional approach, however, we cannot deal with physically implausible but artistically desirable configurations.

In applying our simulation model to artistic fire effects, most of the implausible cases happen when the flow does not follow the physical law. For example, fire flames may be desired to go inside from all four sides of the image for image border decoration. In 2D, since z-directional flow does not exist, such a converging flow has no exit. This makes the flow implausible, resulting in little inward flow as shown in Fig. 1(a). To resolve this problem, we add an additional velocity divergence $q$ as

$$\nabla^2 P = \frac{\rho}{\mu} (\nabla \cdot \mathbf{u} - q), \quad (5)$$

which serves as a source and a sink for the flow. We use $q$ to control the compression/expansion amount and define it based on the buoyancy directional field $\mathbf{b}$. With a constant value $b$, which is 0.1 in our experiments, we have

$$q = b \nabla \cdot \mathbf{b}, \quad (6)$$

Figure 1: Modifications of 2D simulation: (a) no modification, (b) modification with additional divergence but without adaptive buoyancy, (c) both with additional divergence and adaptive buoyancy. (a) is not burning well due to the implausible flow, while (b) and (c) burn well with vorticities and reduced vorticities, respectively.
which means \( q \) is 0 for the region having divergence free buoyancy directions, positive for regions working as sources, and negative for sink regions. The absolute value of \( q \) should be large in regions with rapid changes of the directional field. With this modification, inward flames can be generated, as shown in Fig. 1(b).

In addition to the sink/source effect, we can optionally reduce flame merging, depending on the user preference. Compressibility tends to yield a flame merging effect with vorticities as shown in Fig. 1(b). To reduce this effect, we can increase the buoyancy force proportional to the difference between velocity and buoyancy directions. Fig. 1(c) shows the result of this modification.

5. 2D-Guided 3D Simulation

To generate high-quality image effects with user control, we propose a new simulation strategy, 2D-guided 3D simulation (Fig. 2). We first allow the user to run interactive-time 2D simulation and to choose a frame to be used as the final effect. Then, we rewind the simulated time by a few seconds, extend 2D data to volumetric data, and run the 3D simulation only for the short rewound simulated time. To make the final 3D fire look similar to the selected 2D fire, we guide the 3D simulation using 2D simulation frames. As a result, the final simulated effect contains 3D volumetric detailed patterns resembling the selected 2D simulation frame. This approach enables interactivity with preview and needs less computation time than full 3D simulation.

5.1. Initialization of 3D volumetric field from 2D field

The initial volumetric field should be built before the first 3D simulation step is taken. For this initialization, 3D distributions of fuel, temperature, and velocity should be derived from the 2D simulation result. Although this appears to be an ill-posed problem that could be addressed by a synthesis method, we obtain a simple solution by recognizing that the initialized 3D flow should be similar to 2D. We simply copy 2D data along the \( z \)-axis. Volumetric fluctuation along the \( z \)-axis is not needed for initialization as 3D simulation will develop volumetric flow after some simulated time anyway.

5.2. Flow conversion for guided 3D with 2D constraint

To guide 3D simulation, we impose the 2D result onto the \( z = 0 \) plane in the 3D grid as a constraint. This allows the 3D simulation to develop volumetric patterns almost freely, while maintaining the fire shape similar to the 2D result.

Let \( S_{2D} \) be the plane at \( z = 0 \) in our 3D grid on which the 2D grid used for 2D simulation has been placed. To constrain 3D simulation by 2D simulation results, for explicit steps such as velocity and temperature advections, the 3D simulation result on \( S_{2D} \) is simply overwritten by the corresponding 2D simulation result after each advection. However, for pressure projection and velocity update steps for incompressibility, this may not be sufficient since the incompressibility will no longer be true unless \( \partial w/\partial z = -q \) on \( S_{2D} \), where \( w \) is the \( z \) component of velocity \( \mathbf{u} = (u, v, w) \). Therefore, we modify the pressure projection so that, on \( S_{2D} \), 2D velocities \((u, v)\) are not updated by the projection, and \( \partial w/\partial z = -q \) is enforced, with the following projection equation

\[
\frac{\partial w}{\partial z} = -q \quad \text{on } S_{2D}.
\]

\[
\nabla^2 P = \frac{1}{\rho} (\nabla \cdot \mathbf{u}) \quad \text{otherwise}.
\]

This makes the projection matrix slightly more sparse, since the off-diagonal terms that correspond to \( \partial P/\partial x \) and \( \partial P/\partial y \) are removed. The velocity update step becomes

\[
u^{n+1} = \nu + \frac{\Delta t}{\rho} \frac{\partial w}{\partial z} \quad \text{on } S_{2D}.
\]

\[
u^{n+1} = u + \frac{1}{\rho} \nabla P \quad \text{otherwise}.
\]

This way, we can naturally generate a 3D flow that corresponds to the artificial divergence \( q \) in the 2D simulation. Note that \( w \) can be any value on the 2D slice \( S_{2D} \), and therefore, flows can dynamically pass through \( S_{2D} \). As shown in

\[\text{Figure 2: Overall process of our 2D-guided 3D simulation}\]
Fig. 3, this constraint converts 2D divergence to z-directional flow more accurately than simply applying pressure projection without constraints.

5.3. Boundary conditions

To make the 2D guidance of 3D simulation effective, the behaviors of 2D and 3D simulations should be similar. In that sense, our 2D simulation modification (Sec. 4) should be reproduced in 3D. Whereas the z-directional flow does exist in 3D, this is not the case for the 2D simulation. We use two boundary conditions for 3D simulation, wall boundary and open boundary, which are applied to the front and back faces (slices) of the 3D grid. Our simulation modification strategy in 3D to handle physically implausible flows depends on the boundary condition.

In 3D, with z-directional flow, expansion/compression (2D source/sink) can be converted to flows out or in along the z direction. This implies that physically implausible flow may exit or enter through the z-directional boundary faces, both front and back. Consequently, we normally do not need divergence with the open boundary option. With the wall boundary, however, flow will be bounced back at the front and back faces, containing the 3D fire flow between the two walls. In this case, similar to the 2D case, we add additional divergence on the wall faces to allow the sourcing or sinking flux. On the other hand, adaptive buoyancy for reducing the flame merging effect can be applied in exactly the same way as 2D regardless of the boundary condition.

We have also observed that 3D simulation with open boundary is not similar to 2D simulation as 3D simulation has z-directional flow moving freely. For example, if fire is raised away from fuel (Fig. 4), 3D simulation with open boundary works drastically differently from 2D. Flows entering from open boundaries always accelerate the flow, making the flame grow further (Fig. 4(b)). In contrast, 3D simulation with wall boundary works similarly to 2D simulation, so we can always effectively perform 2D-guided 3D simulation with this option. Fig. 4(c) shows that wall boundary condition does not suffer from the accelerated flow.

However, we note that wall boundary condition is not always required. In the case when the buoyancy is tangential to the fuel direction (Fig. 5), 3D simulation with open boundary is in fact guidable as shown in Figs. 5(c) and 5(d). In this case, if wall boundary condition is used, as shown in Fig. 5(e), guided result looks more similar to 2D.

5.4. Acceleration

After evaluating a wide range of acceleration approaches in space and time, such as adaptive subdivision and multiscales, we resort to two approaches: adaptive time stepping and noisy initialization. Both approaches share the same goal of reducing the total number of frames processed by computationally expensive 3D simulation. This is feasible because our goal is to obtain only one still-frame instead of multiple full frames for animation.

For adaptive time stepping, we use a larger time step at the beginning of the guided 3D simulation, and refine the time step as the target frame is approached. We also experimented with spatially adaptive methods, using lower resolution grids at the beginning for various combinations of projection and advection steps, hoping for fast population of the volumetric field with large features. These spatial approaches, however, generate fuzzy patterns instead of large features because of different levels of detail with 2D simulation, and as a result always diminished the final quality in some degree. In contrast, coarsening the initial time step up to 4∆t did not result in loss of visual detail. In our experiments with the results in Sec. 6, we used the time step 4∆t for the initial 80% of the total simulated time and 2∆t for 80% of the remaining time, reducing the total number of simulation frames to 32% of non-adaptive time stepping.

Noisy initialization is an approach for reducing the re-wound simulated time. In our method, volumetric patterns are generated in 3D as the simulated time goes by. Typically, about two seconds of simulated time are required until flow...
Figure 5: Comparison of various simulation options: We used $1728 \times 2240 \times 64$ grid resolution, 120 frame per second $\Delta t$ (time step), 1m of world scale, and 600th target frame as the goal. For the 2D-guided 3D simulation, we applied 240 frames (2 seconds) rewind for the basic setting, and 120 frames (1 second) rewind for initial noise injection. For adaptive time stepping in (d), we used three levels of $\Delta t$, i.e., $4\Delta t$ for the first half of frames, $2\Delta t$ for the next quarter, and $\Delta t$ for the remaining quarter of frames. (e) is an example of using boundary condition that does not allow flow across front and back boundaries, where consequently the result looks more similar to 2D. (e) also has a shorter rewound time with initial noise injection. The result is obtained faster with slightly reduced detail, demonstrating the cost-quality trade off.

6. Experimental Results

Our new simulation method can generate various kinds of fire effects with simple and intuitive user input that consists of an initial fire region and flow direction. The initial fire region can be automatically determined from the input image, e.g., image border, regions having specific color, or edges. It can also be specified by the user as a mask image. Similarly, the flame direction can be guided easily by a global direction, a flow field constructed from the input image, or a user defined flow field. In addition, we provide several parameters related to simulation and rendering, to fulfill different user preferences for various fire styles.

6.1. Simulation parameters

Major parameters for our 2D fire simulation are the simulation world scale, fuel, and flame direction. These pa-
6.2. Fire rendering

We render a fire image effect using the 2D or 3D temperature fields generated by the fire simulation. For fire colors, we use Planckian locus with black-body assumption, which defines color chromaticity (xy values in CIE xyY color space) from temperature. In our implementation, we use the approximated method of Kang et al. [KMH+02]. The brightness of fire is also correlated with the temperature, based on Rayleigh-Jeans law, where we ignore wavelength differences because the temperature range of fire is relatively small. This brightness is applied as an opacity value to introduce the fading effect of fire. In addition, we give two gamma values, applying gamma correction to color and brightness, for additional variations on the look of fire. Gamma correction of color changes the fire color to be biased toward reddish or yellowish colors, while gamma correction of brightness controls the degree of rapidness of brightness changes.

For a 2D simulation result, fire is visualized by determining the color of each pixel. For a 3D simulation result, we use volumetric accumulation. We first compute the color and opacity values for all voxels in the 3D temperature field. Then, the final fire image is obtained by weighted averaging the color values along the z direction with the opacity values as the weights. When we compose a fire effect with an image, we can optionally apply a bloom effect to the image before composition, in order to roughly simulate the lighting effect of the fire onto the image.

6.3. Fire effect results

Fig. 8 shows various examples of fire effect. Our method generates still fire images in a relatively short time (Table 1). Our simulation domain sizes are one to three meters large, for all results in Fig. 8. Each result has flame details interacting with themselves and decorating photographs in an harmonized way. Among various kinds of fire effects, we mainly focus on the effects for shape generation, because they are the most widely used case and have less rendering issues, such as harmonization with input images.

Fig. 8(a) shows fire letters with inside burning option and upward buoyancy. We could generate natural and plentiful levels of detail with high-resolution of $4224 \times 1408 \times 64$ grid, which was possible in few hours with our 2D-based 3D still-frame simulation. This image is obtained by first running a full-framed 2D simulation for 10 seconds so that flames are raised and developed (this took 1,586 seconds of computation), rewinding for 1.5 seconds, and then running a 3D simulation guided by 2D simulation results with our acceleration technique. Simulating full frames for 10 seconds in 3D might take more than a day, while our 2D-guided 3D simulation took about 3 hours.

Figs. 8(b) to (t) present various kinds of shape generation with fire based on input images, having fuels following edge...
boundaries. They were guided with different buoyancy directions based on the image content. We used the tangential direction for the fuel in Figs. 8(h), 8(m), and 8(r), and various global directions for others. We can generate different fire effects from the same input image using different flows, as shown in Figs. 8(q) and 8(r) and Figs. 8(s) and 8(t).

As the appropriate flow directions for fire effects usually need several trials, our 2D-guided 3D still-frame simulation method would greatly reduce the user interaction time by providing early feedback. We can try fast 2D simulations with several buoyancy directions, and perform 3D simulation only for a short rewound simulated time to produce volumetric version of the most pleasing 2D result. This interactive workflow has become possible because our system is

Figure 8: Various results with different settings: (a) extremely high-resolution result of $4096 \times 1024 \times 64$ grid for burning letters, (b)-(t) various fire shape generation by different buoyancy guides with fuel injection along the edges, (u)-(v) fire image frames with different shapes and buoyancy directions, (w)-(y) burning objects, (z) a burning phrase with fuel at letter boundaries ($4480 \times 512 \times 32$ grid). These examples demonstrate the benefits of our method in effective and efficient generation of still-frame fire effects. To obtain the results, we determined the most proper settings with several trials using fast 2D simulation, and then performed 3D simulation only for a short rewound simulated time towards the final volumetric results.
produce fire-like shapes, but again suffer from a difficulty in depicting natural flow directions (Figs. 9(b) and 9(c)). To alleviate these artifacts, a user can adopt more flame-like input shapes, but it would still generate a textured fire frame, rather than a burning frame, even after a significant amount of user effort.

A popular procedural method in real time graphics is to splat large numbers of fire-textured particles [Fer04]. These procedural methods generate detailed fire appearances in real time. However, a procedural method lacks a number of important effects that exist in simulation methods. There are no laminar flow patterns or transitions to turbulent flow (Figs. 6 and 8), and the fire detail spectrum is concentrated on the typical range depending on the particle size and the fire texture details. In contrast, simulation methods generate a full range of structural details such as variously sized vortices that are starting, fully developing, or decaying, and all of these details are naturally assembled together. We believe that simulation is a crucial step in generating high quality effects even for still images.

7. Conclusion and Future Work

We have shown that a variety of artistic fire effects can be generated using fire simulation. To guide flames with any desirable flows, even in physically implausible ways, we proposed simulation modifications, such as velocity divergences and adaptive buoyancy. To provide quick turnaround time for fire design as an image effect, we developed new simulation methods for still frames that give interactivity to users with 2D simulation and reduce total computation time from a full day to hours for high resolution, and from hours to minutes for low resolution cases. By taking adaptive strides in time and adding noise, simulation produces nice results as the target frame is approached, and hence is optimized for a single frame image effect rather than a full frame animation. To obtain various fire styles, we experimented with different control parameters, such as world scale, fuel amount, and buoyancy adjustment.

7.1. Discussion

The example in Fig. 9(a) is an artist’s work. Although of high quality, this work lacks the impression of fuel burns, heat rises, and various scales of turbulent structures. In addition, it requires artistic skills. Texture synthesis methods can produce fire-like shapes, but again suffer from a difficulty specifically designed for still-frame image effects, while traditional full-frame 3D simulation can hardly provide interactive responses.

Figs. 8(u) to 8(y) show fire effects rendered together with input images. Fig. 8(u) is a fire frame having fuels following the heart shape with a buoyancy direction tangential to the fuel direction, while Fig. 8(v) has fuels following the rectangular shape and inward buoyancy direction. Both can nicely decorate input images with impressive fire frames. Figs. 8(w) and 8(x) are burning guitars, injecting fuels following edges with upward buoyancy direction. Using our system, we could generate natural flames reflecting different shapes of guitars without burning real objects. Fig. 8(y) creates a Phoenix image by burning bird’s wings to fire.

Fig. 8(z) shows another fire letters with upward buoyancy. The resolution of the simulation grid is 4480 × 512 × 32, and we inject fuel following the boundary of letters, unlike Fig. 8(a) burning inside. With our method, sentences, such as wise sayings, could be conveyed in a visually striking way.

### 7.2. Future work

In the case that some unique volumetric patterns can be obtained only by 3D simulation, our 2D-guided 3D simulation approach would fail to produce such patterns. For example, with a lot of fuel injected in 3D space, 3D simulation will proceed quite differently from the 2D case and will generate details with cellular patterns which cannot be obtained by 2D simulation. It would be an important future work to find an effective guiding method for this case.

Our guided simulation can be improved by developing strategies for automatic selection of the final frame during 2D simulation, which is currently done manually by the user.
Developing specific criteria to measure the quality of fire effects appears to be interesting future work.

Physical interaction between fire and a photograph when they are composed with each other can be another interesting area for future work. Photograph charting and warping by heat, as well as relighting with fire, will be yet another interesting interaction to explore.

Image effects can appeal to a large number of end users as they are relatively easy to create, edit, and have multiple varied uses. Using simulation for image effects could be not only an interesting topic but also can be meaningful for the expansion of the research field. We believe and advocate that simulation-based image effects is an exciting new application of simulation methods. For future work to continue this study, we can think of extensions of our approach to other types of simulations, to produce a variety of image effects, and furthermore, to create new video effects.

Acknowledgements

We thank Daichi Ito for Fig. 9(a) and helpful comments, and Flickr users for sharing their images under a Creative Commons license. This research was supported by Basic Science Research Program of NRF (2013-011692, 2012-0008835).

References


