Pacific Graphics (2014)
J. Keyser, Y. J. Kim, and P. Wonka (Editors)

Data-Driven Fire Synthesis and Design

Sai-Keung Wong Tse-Ching Chang Tan-Chi Ho Jung-Hong Chuang

National Chiao Tung University, Taiwan

Abstract

We present a data-driven synthesis approach to design and animate fires in desired shapes and motions. At the preprocessing stage, our system simulates a set of basis fires under specific simulation configurations and stores these basis fires as pathlines in a database. At the design stage, a user sketches a sequence of curves to design the desired shapes of target fires. Then, we compute a subset of basis fires to fit the curves. After that the target fires are synthesized by combining the basis fires. As our method generates target fires along the user sketched curves, our approach enables users to design the fire shapes in an intuitive manner. Experimental results show that our approach can synthesize fires in desired shapes and motions.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

In computer games and movies, it may be desirable to simulate fluid that meets the artistic requirements, e.g., animating fluid in a desired shape. One kind of the methods is to tune the parameter values of the physics based simulators to achieve the desired results. But it would be time consuming and too tedious to do so as the parameter space is large. Thus, there have been techniques which aim for animating fluid in a controllable manner. For example, force fields are designed to guide fluid to move in the simulation space. However, these methods still need users to tune the parameters so as to match the user's desired results. Due to the non-linear property of physics based simulation, a small change of parameter values may lead to a large change in the resulting simulation, leading to difficulty in using these methods.

In this paper, we present a data-driven synthesis approach to synthesize fires whose shapes match with the user sketched curves. At the preprocessing stage, our system simulates a set of basis fires by using a physics based fire simulator. Each basis fire is simulated under a specific simulation condition and the pathlines of the particles are collected in a database. For examples, the basis fires include fires moving to the left or right as well as fires with vortices. In the design stage, a user sketches a set of curves for designing the desired shapes of target fires. Then, our system computes a subset of basis fires to fit the curves. After that the target fires are synthesized by combining the basis fires with appropriate weights.

Our contributions include: 1) A data-driven synthesis method is proposed to synthesize fires that fit the user sketched curves with velocity information; and 2) we adopt a flow graph [ZCM11] to animate the synthesized fires with changing motion in a long lasting manner. The major difference between our method and the method proposed in [ZCM11] is that our method enables a user to use curves to design the shapes of target fires.

2. Related Work

We review the previous works on fluid simulation, fluid control, and fluid synthesis techniques that are closely related to our method.

Fluid simulation. Stam [Sta99] utilized Eulerian grids to solve the Navier Stokes equations. Nguyen et al. [NFJ02] integrated the fire combustion rules to simulate fire. Instead of using grids, Müller et al. [MCG03] adopted the smoothed particle hydrodynamics (SPH) to solve the Navier Stokes equations. The major advantage of the SPH based techniques is that the computation is carried out for the particles but not for the entire simulation space. Goswami et al. [GSSP10] adopted a Z-indexing scheme to handle the particle neighborhood search.

DOI: 10.2312/pgs.20141254

Fluid control. Fluid control is important in computer games, movies and animation. Designers can simulate fluid with control in an artistic way to enhance the attractiveness of fluid motion. Treuille et al. [TMPS03] proposed a gradient-based control method to match the fluid state with the state of target keyframes. McNamara et al. [MTPS04] accelerated the method proposed in [TMPS03] by using an adjoint method so that the cost of optimization computation is reduced.

Dobashi et al. [DKNY08] proposed a feedback control method to control the vapor and heat for cloud simulation. Thurey et al. [TKPR09] used control particles to attract the fluid to flow along with them. Hong et al. [HZQW10] adopted a closest-point method to compute the blue core of fire so that the fire is controlled to burn on a model surface. Lever et al. [LK12] mapped force textures onto model surfaces to control the fire appearance.

Fluid Synthesis. Kim et al. [KTJG08] proposed a wavelet method which enables a user to add details to existing fluid simulations as a post-process. Huang et al. [HMK11] proposed to preview fluid simulation on lower resolution grids and then the results are re-targeted to higher resolution grids. Sato et al. [SMDY12] proposed a data-driven synthesis method to make the fire simulation with low resolution but details were preserved. The precomputed simulation data are collected in high resolution.

Kovar et al. [KGP02] proposed *motion graphs* for animation synthesis to animate characters. Zhang et al. [ZCM11] proposed a modified motion graph called *flow graph* to animate fires. The method samples and stores the physical data along pathlines of particles. Each node of the flow graph represents the entire pathlines which start at the same time step. The edges of the flow graph connect two nodes which have similar physics properties so that transition is possible between these two nodes. The flow graph is constructed at the preprocessing stage. By traversing the flow graph via the edges, a fire animation can be synthesized.

3. Method Overview

We call a fire that will be synthesized as *target fire*. A target fire is synthesized within a time interval which is uniformly sampled as a sequence of time instances. Thus, we use a set of curves to design the shape of a target fire at each time instance. Our framework has four stages: 1) construction of the database of basis fires, 2) fire design, 3) curve fitting and 4) fire synthesis and animation. To compute the database of basis fires, we sample data from the volume data of grid-based fire simulations under specific configurations. A basis fire is constructed according to the simulation data (e.g., velocities and temperatures) for each configuration. In the fire design stage, a user inputs a sequence of curves. The curves describe the desired shapes of a target fire at different time instances. The temperature distribution of the target fire is also specified. In the fitting process, a least square fitting scheme is employed to compute a subset of the basis fires to fit the curves. We also construct a flow graph so that a long lasting animation of fire can be produced.

4. Database Construction

To construct the database of basis fires, we employ a grid-based technique to simulate fire due to its accuracy and simplicity. We use particles to trace the volume data and store the simulation data such as pathlines and velocities of the particles.

4.1. Fire Simulation

We simulate fire based on the Navier-Stokes Equations and combustion rules. The Navier-Stokes Equations are defined by Eq.1 and Eq.2 as follows:

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

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

where t, $\mathbf{u}$, p, ρ and $\mathbf{f}$ are time, velocity, pressure, density and external force, respectively. The combustion rules in [ZCM11] are simplified into two rules which update fuel and temperature. If the fuel is turned into soot, it will not take part in the reaction of combustion. Thus, we do not take soot into simulation for simplicity. The simplified combustion rules are given as follows:

$$\frac{\partial F}{\partial t} = -(\mathbf{u}\cdot\nabla)F - rF, \tag{3}$$

$$\frac{\partial K}{\partial t} = -(\mathbf{u}\cdot\nabla)T - c\left(\frac{K-K_0}{K_{max}-K_0}\right)^4 + k_K rF, \tag{4}$$

where F is fuel, K is temperature, K_{max} is maximum temperature, K_0 is ambient temperature, r is reaction rate, c is cooling rate and k_K is transfer rate of temperature. The terms on the right hand side of Eq. 3 are the convection of fuel and the consumption of fuel due to combustion reaction. The terms on the right hand side of Eq. 4 are the convection of temperature, the cooling of temperature, and the gain of temperature due to combustion reaction.

The right most term $\mathbf{f}$ in Eq.1 is the net force of all the external forces, including buoyant force, wind force, and paddle wheel force [FSJ01]. We can configure the external forces to generate fires with different features. For example, to generate a fire under wind blowing in a direction, we can tune the wind force to blow the fire in that direction. The paddle wheel force adds vorticity to fluid and it is computed by Eq. 5 as follows:

$$\mathbf{f}_v = \varepsilon h\left(\frac{\nabla|\omega|}{|\nabla|\omega||}\times\omega\right), \varepsilon > 0, \tag{5}$$

where $\omega = \bigtriangledown \times \mathbf{u}$, h is the grid size and ε is a scaling factor for controlling the magnitude of the paddle wheel force. As $\mathbf{f}_v$ depends on the spatial discretization, the fluid vorticity is preserved. By assigning ε with different values, we can simulate fires with different scales of vortices. We adopted a grid-based approach to implement the technique on a GPU platform using NVIDIA's CUDA framework.

4.2. Basis Fires

In the following, we decide the set of basis fires. In the normal condition, fires burn upwards (i.e., $+Y$ direction) naturally due to the convection of air and buoyancy. Thus, an important criterion for the set of basis fires is that the combination of them spans the x-z plane. Furthermore, vortices also appear in fires that make the shapes of fires changing in a rich pattern. In general, fires would have small-scale vortices under the normal condition. Sometimes, we would also like to generate fires with large-scale vortices. Therefore, the basis fires are collected under the following three configurations: 1) wind blowing along the $+X$ direction, 2) wind blowing along the $+Z$ direction, and 3) fire with large-scale vortices. Examples are shown in Fig. 1. To generate fire with large-scale vortices, we can set ε in Eq. 5 to achieve the goal. By combing the first and second basis fires, we can synthesize fires that span the x-z plane. The third basis fire enables us to produce large-scale vortices in the synthesized fires.

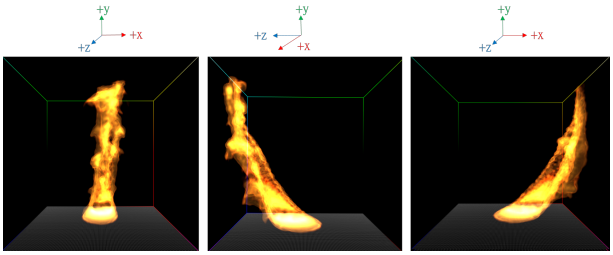

Figure 1: *The basis fires in our database. Left: fire with a large-scale vortex; Middle: wind blowing in $+Z$ direction; Right: wind blowing in $+X$ direction. The coordinate system is shown at the top of each image.*

4.3. Data Sampling Using Particle Tracing

We use particles to sample the volume data while the grid based simulation is executed. The data, such as the velocities and temperatures of the particles, are stored in the database. In this way, we do not need to store the entire volume data and thus save memory space.

We seed a group of particles in the emitting region Ω_B of the fire at each time step. Ω_B is regularly divided into a grid. The particles move according to the velocity field in the grid during their lifetimes. For each particle, the simulation data are recorded along the trajectory of the particle at each time step. When the lifetime of a particle is over, we obtain a collection of velocities and temperatures along the pathline of the particle.

Let's consider the i-th basis fire. Denote that γ is the emitting time, $\mathbf{x}$ is the emitting location and τ is the elapsed time for a particle. For simplicity, γ and τ are measured in the number of simulation time steps. Let $p_B^i(\gamma,\mathbf{x},\tau)$ be the simulation data collected for the particle at the elapsed time τ. $p_B^i(\gamma,\mathbf{x},\tau)$ is defined as follows:

$$p_B^i(\gamma,\mathbf{x},\tau) = \{\ \mathbf{u}_B^i(\gamma,\mathbf{x},\tau)\ ,\ \kappa_B^i(\gamma,\mathbf{x},\tau)\ \}, \tag{6}$$

where $\mathbf{u}_B^i(\gamma,\mathbf{x},\tau)$ and $\kappa_B^i(\gamma,\mathbf{x},\tau)$ are the velocity and temperature, respectively. In the local time space, the pathline $P_B^i(\gamma,\mathbf{x},t)$ of the particle is given by

$$P_B^i(\gamma,\mathbf{x},t) = \mathbf{x} + \int_0^t \mathbf{u}_B^i(\gamma,\mathbf{x},\tau)d\tau, \tag{7}$$

where $t \in [0,\tau_{max}]$ and τ_{max} is the maximum life time of the particle.

4.4. Database

To reconstruct the volume data of the i-th basis fire at a particular time instance, we need a bunch of pathlines and they are emitted at the same time. Let $b_i(\gamma,t)$ be a bunch of pathlines of a set of particles such that the particles are emitted at the same time γ from region Ω_B. $b_i(\gamma,t)$ is defined as

$$b_i(\gamma,t) = \{\ P_B^i(\gamma,\mathbf{x},t)\ |\ \forall \mathbf{x} \in \Omega_B\ \}, \tag{8}$$

In order to capture the volume data of a basis fire for the entire simulation, we need to store a bunch sequence which contains all the bunches of the basis fire at each simulation time step. Let B_i be a bunch sequence and it is defined as follows:

$$B_i = \{\ b_i(\gamma,t)\ |\ 0 \le \gamma \le t_{max}\ \}, \tag{9}$$

where t_{max} is the maximum number of the simulation steps in the entire simulation of the basis fire. Thus, we can use B_i to reconstruct the volume data of a basis fire at a specific time within $[0,t_{max}]$.

The database Σ_B contains a collection of bunch sequences for all basis fires. Thus, we have

$$\Sigma_B = \{\ B_i\ |\ i = 1,2,...,M_B\ \}, \tag{10}$$

where M_B is the number of basis fires.

5. Target Fire Design

A target fire is synthesized over a time interval and time instances are sampled uniformly within the time interval. The goal of fire design is to create a set of curves at the time instances for defining the shapes, velocities and temperature distributions of target fires. The set of curves at each time instance will be mapped to a subset of bunch sequences in the fitting process.

5.1. Fire Shape Design

We use a curve to represent a part of the desired shape of a target fire, as shown in Fig. 2. A curve consists of a sequence of nodes $\{\mathbf{n}(0),\mathbf{n}(1),...,\mathbf{n}(\tau_{max})\}$. Thus, we can treat a curve as a pathline of a particle as the particle moves along the curve continuously. To do so, we associate each node with a velocity to produce a *target pathline*. The velocity $\mathbf{u}_T(\tau)$ at node $\mathbf{n}(\tau)$ is computed as follows:

$$\mathbf{u}_T(\tau) = \frac{\mathbf{n}(\tau+1)-\mathbf{n}(\tau)}{\Delta t}, \tag{11}$$

where Δt is the time step size used in animation. In our case, we set $\Delta t = 1$. Thus, if a particle moves along the curve based on the velocities associated with the nodes, the particle traces along the curve. By setting the nodes to different locations on the curve, we can obtain a different sequence of velocities along the curve.

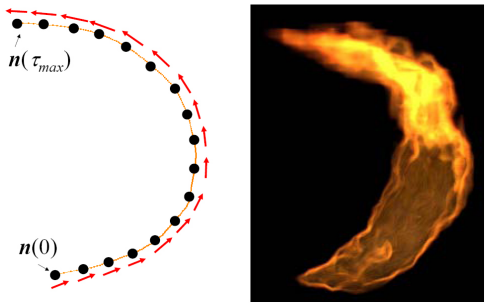

Figure 2: *Target fire design. Left: a curve used for defining the desired shape of a target fire. The red arrows indicate the velocities associated with the nodes. Right: the synthesized target fire is generated according to the curve.*

As the shape of a target fire occupies a volume, we use a bunch of curves to model its shape. However, it would be tedious to edit all such curves. Instead, once we have created a curve, the curve is duplicated for multiple times to obtain a set of curves. A noise (e.g., guassian) is added to the curves so that they have variant shapes. These curves are placed close to each other. Each curve is then converted into a target pathline. In the fitting process, the target pathlines of the target fire are fitted by the pathlines of the basis fires.

5.2. Temperature Design

The fire temperature changes dynamically from time to time. The temperature distribution within a fire volume is too complex to create by hands. Therefore, we propose to assign temperature templates to produce the temperature distribution of a target fire. A temperature template contains the temperature distribution for a group of particles that are emitted at the same time step. A set of temperature templates can be obtained while we construct the database of basis fires. While the pathlines of the particles are being collected (see Section 4.3), the temperature data along the pathlines are also recorded. Thus, a temperature template contains the pathlines and the temperature data along the pathlines. We can collect a set of templates at consecutive emitting time instances and use them to animate a target fire with dynamically changing temperatures.

Now, we have a set of temperature templates. A user can select a temperature template and our system maps the target pathlines of the target fire to the temperature template. We build adjacency graphs for the emitting locations of pathlines and then apply an adjacency graph mapping to achieve spatial coherence.

An adjacency graph $G = (V,E)$ consists of a set of nodes V and a set of edges E. Each node represents the emitting location of a pathline. An edge connects two nodes if the distance of their emitting locations is smaller than a user defined threshold ε_d. We construct two adjacency graphs $G^{temperature}$ and G^{target} for the pathlines of the temperature template and the target pathlines, respectively.

The adjacency graph mapping scheme maps each pathline of the temperature template to a target pathline. First, we align both graphs G^{target} and $G^{temperature}$ by superimposing them. Then, we map a node of G^{target} to a node of $G^{temperature}$. These two nodes are selected as the centroids of the two graphs. After that we start from these two nodes and apply a breadth-first search to map the remaining nodes of G^{target} to those of $G^{temperature}$. After performing the mapping scheme, each node $\mathbf{n}(\tau)$ of a target pathline is associated with a temperature $\kappa(\tau)$. Thus, the data at each node $\mathbf{n}(\tau)$ can be denoted as $p_T(\mathbf{x},\tau) = \{\mathbf{u}_T(\mathbf{x},\tau),\kappa_T(\mathbf{x},\tau)\}$.

6. Fire Synthesis Using Fitting

Our approach synthesizes a fire by combining the basis fires with suitable weights. Similar to the mapping between G^{target} and $G^{temperature}$, we also consider the spatial coherence between the basis pathlines and the target pathlines in the fitting process. We construct an adjacency graph G^{basis}. A node in G^{basis} represents the emitting location of a basis pathline of a basis fire. Our task is to compute a surjective map which maps each node of G^{target} to a node of G^{basis}. To construct the surjective map, we superimpose the two graphs and align them at their centroids. Then, the two nodes at the centroids are mapped. After that a breadth-first search approach is adopted to map the remaining nodes of G^{target} to the nodes of G^{basis}.

6.1. Velocity Fitting

The velocity fitting method is to fit the target pathlines by using a linear combination of basis pathlines of one or more than one basis fire. If we fit a target pathline as a whole, the fitting error could be too large due to the variation in the

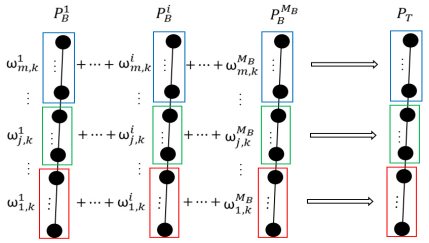

Figure 3: *The segments of basis fires in the same color are combined to obtain the corresponding target segments on the right hand side.*

pathline's shape. Thus, we divide each target pathline into segments. A pathline $p_T^k(\mathbf{x},\tau)$ at the k-th time instance is divided into m segments $p_T^k(\mathbf{x},I_1),...,p_T^k(\mathbf{x},I_j)),p_T^k(\mathbf{x},I_m))$, where I_j are intervals, the interval length is $s=\tau_{max}/m$. Therefore, we formulate the fitting problem for the piecewise segments. We have the following approximation for the j-th segment:

$$\mathbf{u}_T^k(\mathbf{x},\tau) \approx \sum_{i=1}^{M_B} \omega_{j,k}^i \mathbf{u}_B^i(\gamma,\mathbf{x},\tau), \forall \tau \in I_j \qquad (12)$$

where $\mathbf{u}_T^k(\mathbf{x},\tau)$ is the velocity associated with the segment of $p_T^k(\mathbf{x},\tau)$, $\mathbf{u}_B^i(\gamma,\mathbf{x},\tau)$ is the velocity associated with the segment of the i-th basis pathline, and $\omega_{j,k}^i$ are the weights. An example is illustrated in Fig. 3.

On the one hand, we want to synthesize fires in similar shape to the curves. On the other hand, the synthesized fires should be animated vividly. Thus, we adopt a velocity fitting scheme for the piecewise segments and formulate the scheme as a least square fitting problem. Our goal is to minimize the total velocity change between the target pathlines and pathlines of basis fires. The result of the least square fitting problem is a set of weights associated with the basis fires. Thus, we solve the following least square fitting problem

$$\arg\min_{\omega_{j,k}^i} \sum_{\tau\in I_j} \| \mathbf{u}_T^k(\mathbf{x},\tau) - \sum_{i=1}^{M_B} \omega_{j,k}^i \mathbf{u}_B^i(\gamma,\mathbf{x},\tau) \|^2. \qquad (13)$$

We used the armadillo C++ linear algebra library to solve the least square problem.

7. Fire Animation

If we simply use the fitting results to animate the target fire, the animation can be played at most for τ_{max} time steps. We are inspired by the flow graph [ZCM11] and develop a similar technique to produce a long lasting animation of synthesized fire. Also, we employ linear interpolation for the synthesized target fire to transit from one time instance to the next time instance. To achieve fast rendering, we use a simple ray casting technique to render the fire according to the temperature data of grids. We render the slices of grid data from back to front. The transfer function for the colors and opacities is precomputed and stored in a color texture, as shown in Fig. 4.

Figure 4: *The color table.*

8. Results

We conducted experiments on a PC with an Intel E3-1230 V2 3.3GHz CPU (8 cores) and an NVIDIA GTX 770 GPU. A single thread of CPU performed the fitting process and multi-threads CPU performed the flow graph traversal. The simulation grid dimension was 128^3 in all our examples. The runtime cost for traversing the flow graph depends on the total number of emitted particles. For higher number of emitted particles, the performance would decrease. Table 1 shows the performance statistics.

	Pathlines /Bunch	Piecewise Segment Length	# time instances	Synthesis time (sec)	#Emitted particles (k)	Runtime Cost (ms)
'X'	2016	10	9	8.0	118.9	16.5
'8'	2104	3	9	11.2	126.2	16.9
Heart	648	10	24	1.2	38.9	13.6
PG	3400	3	9	19.0	200.6	19.5
2014	1868	5	9	8.0	110.2	15.9
Antlers	3312	6	9	14.0	195.4	19.6

Table 1: *Performance statistics.*

The synthesized fires are shown in Fig. 5 and Fig. 6. Fig. 5(a) shows two symbols 'X' and '8'. In these two examples, two symmetric curves were combined together to design the fire shapes. There are 10 curves in the word "PG2014", as shown in Fig. 5(b). The 'Heart' symbol (Fig. 5(c)) was modeled by two symmetric curves. As our method synthesizes the fires which move along the curves, it is easy to produce the desired shapes of fires. Fig. 6 shows the 'antlers' of a deer animated as fires. There are 3312 pathlines per bunch in this example. The average runtime cost is 19.6 msec per frame. but the synthesis time is 14.0 sec.

9. Conclusion and Future Work

We have proposed a method for designing and synthesizing fire animation. Our synthesis method fits a set of basis fires to the user defined curves. As our method does not employ physical control rules, our method is free from any physical parameter tunings. A user can simply draw curves which represent the shapes of the synthesized fires. Our system can animate the fires in the desired shapes and motions accordingly. Our method has limitations. The synthesized fires look like tube shapes because the particles move along the curves.

Figure 5: *Results for synthesized fires in simple shapes. a) symbols 'X' and '8'. b) There are ten curves used to produce the word 'PG2014'. c) A changing heart shape.*

Figure 6: *The antlers of a deer.*

Our method may not be easy to create fires emitting from a large region. There would be a lot of curves to be edited. The computational cost for solving the least square fitting problem is costly. If the basis fires are changed often, the cost of the total process is quite high. In the future work, we would like to tackle these limitations. A scheme should be developed to enhance the details of synthesized fires.

Acknowledgement

We would like to thank the reviewers for their invaluable comments. This paper was supported in part by Ministry of Science and Technology of ROC (Taiwan) grants under MOST 103-2221-E-009-127 and MOST 103-2221-E-009-122-MY3.

References

[DKNY08] DOBASHI Y., KUSUMOTO K., NISHITA T., YAMAMOTO T.: Feedback control of cumuliform cloud formation based on computational fluid dynamics. In *ACM Transactions on Graphics* (2008), vol. 27, p. 94.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of the 28th annual conference on computer graphics and interactive techniques* (2001), pp. 15–22.

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive sph simulation and rendering on the gpu. In *Proceedings of Symposium on Computer Animation* (2010), pp. 55–64.

[HMK11] HUANG R., MELEK Z., KEYSER J.: Preview-based sampling for controlling gaseous simulations. In *Proceedings of Symposium on Computer Animation* (2011), pp. 177–186.

[HZQW10] HONG Y., ZHU D., QIU X., WANG Z.: Geometry-based control of fire simulation. *The Visual Computer 26*, 9 (2010), 1217–1228.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *ACM SIGGRAPH* (2002), pp. 473–482.

[KTJG08] KIM T., THÜREY N., JAMES D., GROSS M.: Wavelet turbulence for fluid simulation. In *ACM SIGGRAPH* (2008), pp. 50:1–50:6.

[LK12] LEVER J., KOMURA T.: Real-time controllable fire using textured forces. *The Visual Computer 28*, 6-8 (2012), 691–700.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of Symposium on Computer Animation* (2003), pp. 154–159.

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *ACM Transactions On Graphics* (2004), vol. 23, pp. 449–456.

[NFJ02] NGUYEN D. Q., FEDKIW R., JENSEN H. W.: Physically based modeling and animation of fire. *ACM Transactions on Graphics 21*, 3 (2002), 721–728.

[SMDY12] SATO S., MORITA T., DOBASHI Y., YAMAMOTO T.: A data-driven approach for synthesizing high-resolution animation of fire. In *Proceedings of the Digital Production Symposium* (2012), pp. 37–42.

[Sta99] STAM J.: Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), pp. 121–128.

[TKPR09] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. *Graphical Models 71*, 6 (2009), 221–228.

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. In *ACM Transactions on Graphics* (2003), vol. 22, pp. 716–723.

[ZCM11] ZHANG Y., CORREA C. D., MA K.-L.: Graph-based fire synthesis. In *Symposium on Computer Animation* (2011), pp. 187–194.