# Animating Fire with Sound

Jeffrey N. Chadwick Doug L. James

Cornell University

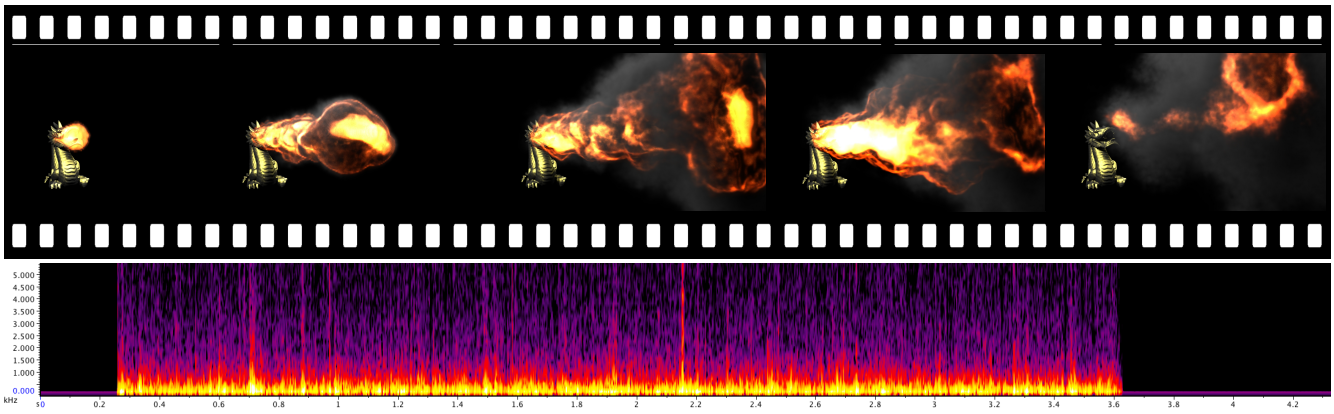


**Figure 1: Fire Sound Synthesis:** *Our method produces the familiar sound of roaring flames synchronized with an underlying low-frequency physically based flame simulation. Additional mid- to high-frequency sound content is synthesized using methods based on spectral bandwidth extension, or sound texture synthesis for user-controlled flame sound styles.*

## Abstract

We propose a practical method for synthesizing plausible fire sounds that are synchronized with physically based fire animations. To enable synthesis of combustion sounds without incurring the cost of time-stepping fluid simulations at audio rates, we decompose our synthesis procedure into two components. First, a low-frequency flame sound is synthesized using a physically based combustion sound model driven with data from a visual flame simulation run at a relatively low temporal sampling rate. Second, we propose two bandwidth extension methods for synthesizing additional high-frequency flame sound content: (1) spectral bandwidth extension which synthesizes higher-frequency noise matching combustion sound spectra from theory and experiment; and (2) data-driven texture synthesis to synthesize high-frequency content based on input flame sound recordings. Various examples and comparisons are presented demonstrating plausible flame sounds, from small candle flames to large flame jets.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation; H.5.5 [Information Systems]: Information Interfaces and Presentation—Sound and Music Computing

**Keywords:** Sound synthesis; combustion; fire; bandwidth extension; texture synthesis

## 1 Introduction

Candle flames, stove top burners and campfires are all familiar combustion phenomena. Larger flame sources such as flamethrowers, burning wreckage and fire-breathing dragons are familiar fixtures in the special effects industry. Due to the unsteady nature of combustion, these structures all tend to behave as noisy sound sources. Physically based fire simulators are capable of producing compelling visual simulations modeling all of these phenomena. Unfortunately, in spite of their ability to produce rich visual behavior, these solvers produce little information suitable for direct synthesis of flame sounds. Recorded combustion sounds can provide compelling auditory feedback, but they can require manual intervention, and can fail to produce realistic synchronized sounds which match visual flame behavior. While physically based sound synthesis methods have been developed for vibrating solid bodies [O'Brien et al. 2001; O'Brien et al. 2002; van den Doel et al. 2001], fracturing solids [Zheng and James 2010], aerodynamic phenomena [Dobashi et al. 2003; Dobashi et al. 2004] and splashing fluids [Zheng and James 2009; Moss et al. 2010], none exist for synthesizing the familiar sound of flames.

In this paper, we present a hybrid method for synthesizing plausible sounds due to combustion phenomena (see Figure 1 for a preview of our results). Rather than building a custom flame solver specifically for sound synthesis, we instead design a sound model which can be driven by data from current physically based animations. Using our sound model, existing simulators can synthesize synchronized sounds. However only low-frequency sounds, such as rumbling from very large flames, can be synthesized in practice for two reasons: (1) time-stepping combustion phenomena at audio rates is impractical due to the high computational costs of 3D flame simulation; and (2) real combustion noise results from complex thermo-acoustics of chemically reacting flows which are unresolved by most flame animations. Sounds recorded in high-speed video experiments (see §6) reveal detailed temporal behavior at a variety of time scales which cannot be resolved by flame solvers run at just graphics rates.

With this in mind, we propose a hybrid technique in which flame

sounds are physically simulated at only a few hundred Hertz, then additional mid- to high-frequency details are synthesized procedurally using one of two approaches:

1. *Spectral bandwidth extension* synthesizes high-frequency power-law noise to extend the frequency response of our otherwise low-frequency sounds. Frequency-domain sound synthesis methods are used to blend synchronized noise, and thereby produce power-law spectra that matches theoretical and experimental models of turbulent flames.

2. *Sound texture synthesis* allows us to synthesize fine-scale sound structure with more interesting temporal details. A modified coarse-to-fine texture synthesis method [Wei and Levoy 2000] is used to coherently synthesize detail on top of the input low-frequency physics-based sound. By varying the flame sounds used for input training data, users can control the style of synthesized sounds, while retaining synchronization with simulated flames.

**Other Related Work:** The most closely related graphics work is the work on aerodynamic sound rendering by Dobashi et al. [2003]. They devise a method for efficiently rendering aerodynamic sounds due to vortex-based noise [Howe 2003], and effectively apply it to swinging clubs and sticks, and whistling wind. Their run-time efficiency is due to the use of precomputed flow noise signals. In a follow-up work [Dobashi et al. 2004], a method is proposed for general aerodynamic flow noise, and several examples are presented which involve sounds produced by turbulent flames. While these examples demonstrate the versatility of the aerodynamic sound model, we point out that the dominant source of combustion sound is not vortex-based noise [Schwarz and Janicka 2009]. For example, results from numerical and laboratory experiments confirm that another source of sound (namely, direct combustion sound; see §2) is the primary contributor to combustion noise, and that aerodynamic noise typically makes a relatively small contribution [Ihme et al. 2009]. In engineering, numerical methods exist for combustion noise based on resolving the thermo-acoustics of chemically reacting flows [Schwarz and Janicka 2009], however the simulation methods (such as large eddy simulation) can be orders of magnitude more expensive than visual flame simulations, which would greatly limit the practicality of fire sound synthesis.

Non-physics-based synthesis of flame sounds synchronized with animation have also been considered, but can lack close synchronization with 3D flame state. Frequency-domain sound synthesis methods, descendant from spectral modeling synthesis [Serra and Smith III 1990], have been used to synthesize the noise-like roar, hiss and crackle of an animated fireplace [Marelli et al. 2010]. Our noise-based bandwidth extension method uses similar frequency-domain noise methods to enhance our low-frequency physics-based sound. It was inspired by (blind) bandwidth extension methods used in the audio community to add high-frequency detail to degraded signals such as digitally encoded audio [Liu et al. 2003; Larsen and Aarts 2004; Annadana et al. 2007]. McDermott et al. [2009] synthesized a fire sound clip using spectral noise, and found that missing temporal structure could be modeled using higher-order statistics. In contrast, we use a spectral noise model and obtain temporal structure from a low-frequency physics-based fire sound.

Sound texture modeling and synthesis provides another way to re-synthesize fire sounds from recordings (see [Strobl et al. 2006] for a recent review), and includes audio generalizations of image texture synthesis [Efros and Leung 1999; Wei and Levoy 2000; Efros and Freeman 2001]. Granular synthesis [Roads 2004] is a classic method for re-synthesizing micro-sound details, but it can be difficult to arrange grains so as to re-synthesize meso-scale temporal structures. Wavelet tree learning [Dubnov et al. 2002] is able to re-synthesize stochastic and quasi-periodic textures effectively, but offers no automatic input control needed for fire sound synthesis. Motion-driven sound synthesis [Cardle et al. 2003] provides a fully automatic control technique wherein a training motion signal w/ sound is segmented and used to map sound onto an input motion signal, e.g., of a 2D car motion. Unfortunately, it does not generalize to 3D fire motions, and our experience with synthesizing short segments of audio (using low-frequency sound segments to index high-frequency content) suffered from obvious granular-synthesis-like artifacts. In contrast, we use a modified texture synthesis approach wherein low-frequency fire sounds seed a coarse-to-fine sound texture synthesis similar to [Wei and Levoy 2000].

## 2 Background

### 2.1 Physically Based Flame Simulation

Our sound synthesis approach is designed to build upon existing flame solvers familiar to the computer graphics community [Nguyen et al. 2002; Hong et al. 2007; Horvath and Geiger 2009]. In particular, our examples are generated using the *Pyro FX* solver and *Flame Solver* from *Side Effects Software*'s *Houdini* 3D animation tools package (`http://www.sidefx.com`). The flow of gaseous fuel and products in a 3D domain is modeled using the Euler equations

$$0 = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p \quad \text{where} \quad \nabla \cdot \mathbf{u} = \phi. \qquad (1)$$

Here $\phi$ is an optional *divergence source* term introduced in the vicinity of combustion to model the effect of expanding gases [Feldman et al. 2003]. A density field is used to govern the introduction, advection and diffusion of smoke [Fedkiw et al. 2001]. A temperature field is also modeled to track the introduction of heat by combustion and to drive forces such as thermal buoyancy [Foster and Metaxas 1997]. Other effects such as vorticity confinement and procedural turbulence may be introduced to increase the liveliness and visual realism in simulated flames [Nguyen et al. 2002; Horvath and Geiger 2009].

Combustion is modeled in [Nguyen et al. 2002] using a *blue core* model. In premixed flames (flames in which reactants are mixed and will burn immediately upon reaching a certain temperature) the blue core specifies a *flame front* interface between unburned fuel, and hot gaseous products produced by rapid combustion as fuel crosses the flame front. Separate sets of incompressible flow equations are used to model the flow of unburned fuel and gaseous products inside and outside of the flame front. The level set method is used to track a moving implicit surface representing the flame front. Voxels outside of the flame front store reaction coordinates which track the time elapsed since the gas stored in each voxel crossed the front. These quantities are combined with a flame temperature profile function to track the temperature of gaseous products outside of the front.

In Houdini's Pyro FX solver, fuel is stored as a volume field along with temperature, velocity, etc. Fuel has an associated ignition temperature $T_I$ and burn rate $b$. At a given time step, if voxel $(i, j, k)$ has fuel concentration $f_{ijk} > 0$ and temperature $T_{ijk} > T_I$ then a quantity of fuel $\Delta f_{ijk} = \Delta t b$ is consumed and removed from voxel $(i, j, k)$. Here $\Delta t$ refers to the simulation time step. Temperature $T_{ijk}$, divergence sourcing $\phi_{ijk}$ and density $\rho_{ijk}$ are modified according to how much fuel is consumed. To make the flames more lively, synthetic turbulence can be injected into the velocity field [Bridson et al. 2007; Kim et al. 2008].

### 2.2 Combustion Sound Generation

In general, sounds produced by combustion phenomena can be expressed as a sum of contributions from multiple sources. A wave equation for combustion sound can be derived which includes contributions from turbulent vortex-based flow noise and direct combustion noise [Chrighton et al. 1992]. Numerical and experimental results in [Ihme et al. 2009] suggest that direct combustion noise is indeed the dominant source of sound from combustion phenomena. The primary quantity of interest in our work is direct combustion noise; namely, noise produced by unsteady density fluctuations resulting from combustion heat release. Under this assumption, a linear wave equation describing combustion sound is given as follows [Chrighton et al. 1992; Rajaram and Lieuwen 2009]:

$$\frac{1}{c_0^2}\frac{\partial^2}{\partial t^2}p - \nabla^2 p = -\frac{\partial}{\partial t}\left(\frac{\rho_0\,(\gamma-1)\,q}{\rho c^2}\right) \tag{2}$$

$p$, $c$, $\rho$ and $\gamma$ refer to the unsteady pressure, speed of sound, density and ratio of specific heats of air, respectively; $c_0$ and $\rho_0$ refer to the ambient speed of sound and density; $\gamma$ is assumed to be independent of temperature, and combustion is assumed to take place at ambient pressure $p_0$. The heat release rate $q$ is the rate at which heat is introduced in to the domain by combustion. The free-space Green's function can be used to solve (2) for the sound pressure [Chrighton et al. 1992],

$$p(\boldsymbol{x},t) = \frac{1}{4\pi}\frac{\gamma-1}{c_0^2}\frac{\partial}{\partial t}\int_{\mathbb{R}^3}\frac{[q]}{\|\boldsymbol{x}-\mathbf{y}\|}d^3\mathbf{y}, \tag{3}$$

where brackets $[\cdot]$ refer to evaluation at $t - \|\boldsymbol{x}-\mathbf{y}\|/c_0$.

### 2.3 Combustion Sound Spectra

Numerous theoretical and experimental investigations into the acoustic power spectrum emitted by flames have been carried out. Abugov and Obrezkov [1978] recorded sounds generated by flames in a propane burner, and observed that in the high frequency region (beyond a certain peak frequency) the power spectrum of acoustic emissions from the flames exhibits a $P(f) \propto f^{-5/2}$ power law. Mathematical derivations by Clavin and Siggia [1991] showed that the $f^{-5/2}$ power law would result if the flame front were subjected to fully developed velocity turbulence that obeyed a Kolmogorov energy spectrum, $E(k) \propto k^{-5/3}$ where $k$ is the wave number. Note that Kolmogorov turbulence models are commonly used in fluid animation [Kim et al. 2008]. Very recent work performed a variety of experiments to study the acoustic emissions of turbulent flames [Rajaram and Lieuwen 2009], and also found that beyond a certain peak frequency the acoustic power spectrum appears to obey a power law $P(f) \propto f^{-\alpha}$, but that fitting this model to experimental data suggested plausible values for $\alpha$ in the range $2.1 < \alpha < 3.4$.

## 3 Low-Frequency Fire Sound Synthesis

### 3.1 Fire Sound Model

Under the premixed flame assumption, unburned gas is consumed and releases heat via combustion very quickly as it crosses the flame front [Nguyen et al. 2002]. Strahle [1972] argued that only velocity fluctuations in the vicinity of the flame front are responsible for significant sound output. To model low-frequency sound output from a flame simulation, we make the assumption that the heat release due to velocity fluctuations in the vicinity of a flame front surface patch $\delta S$ about point $\boldsymbol{x}$ is proportional to the velocity flux across that patch

$$\delta q(\boldsymbol{x}) = \mathbf{u}(\boldsymbol{x}) \cdot \mathbf{n}(\boldsymbol{x})\;\delta S \tag{4}$$

where $\mathbf{n}(\boldsymbol{x})$ is the normal to the surface patch $\delta S$ at point $\boldsymbol{x}$. Using this, the volumetric heat release integral can be rewritten as an integral over the front surface $S$

$$\int_{\mathbb{R}^3} q\; d^3\boldsymbol{x} = \int_S \mathbf{u}\cdot\mathbf{n}\; dS. \tag{5}$$

Intuitively, this model suggests that the rate at which heat is released in to the domain at a point $\boldsymbol{x}$ on the front surface is proportional to the rate at which fuel is carried in to the surface at that point. Relationships between the volume integral of the heat release rate and surface velocity flux integrals of this form occur in the literature [Strahle 1972; Clavin and Siggia 1991; Chrighton et al. 1992]. We ignore time delays and distance attenuation in (3), and also omit constants, since they just provide a fixed scaling to the output sound. Using these simplifications and (5), we can generate low-frequency flame sounds using

$$p(t) = \frac{d}{dt}\int_{S(t)}\mathbf{u}(\boldsymbol{x},t)\cdot\mathbf{n}(\boldsymbol{x},t)\;dS(\boldsymbol{x}) = \frac{d}{dt}I(t) \tag{6}$$

where the flame front $S$, its normal $\mathbf{n}$ and the velocity field $\mathbf{u}$ are all time-dependent quantities. Figure 2 provides a schematic of the geometry used in this sound model.


**Figure 2: Flame front surface:** *Sound is computed by evaluating and differentiating a velocity flux integral $I(t)$ over this dynamic surface.*

### 3.2 Flame Front Estimation

Since flame solvers model the flow of smoke, gaseous products, fuel, etc., the velocity field $\mathbf{u}$ is readily available at any point in the domain (see §2.1). In some solvers, a representation of the flame front may also be available. For example, level set methods track an implicit surface representing the flame front [Nguyen et al. 2002], from which an iso-surface can be extracted using Marching Cubes [Lorenson and Cline 1987] or a similar method.

However, as described in §2.1, not all flame solvers use an explicit representation of the flame front to model combustion. In these cases, we must estimate a representation of the flame front surface which is physically plausible and resolves the turbulent dynamics. Given a fuel-based combustion model, such as the one presented in §2.1, we can define a field $B_{ijk}$ whose contents specify the current rate of fuel consumption in voxel $(i,j,k)$. for example, this information is stored in the "burn" field in Houdini's solver. By extracting a suitable iso-surface, we can estimate a plausible interface separating unburned fuel and burned combustion products.

Extracting an iso-surface directly from data on a modest-resolution voxel grid tends to produce jagged surfaces with non-smooth normal fields. Unfortunately such surfacing artifacts can result in undesirable sound artifacts with our method. To address these problems we use a higher-order cubic interpolation method to reconstruct $S$. Suppose that we wish to build an iso-surface from field $B$, which is discretized on a finite difference grid at points $B_{ijk}$, which corresponds to the value of $B$ at point $(x_i, y_j, z_k)$. In order to recover a smooth front surface $S$ and normal field $\mathbf{n}$ we introduce interpolation functions $\phi_{ijk}(\boldsymbol{x})$ and define a new field $\tilde{B}$ as follows:

$$\tilde{B}(\boldsymbol{x}) = \sum_{i,j,k}\phi_{ijk}(\boldsymbol{x})B_{ijk} \quad \text{where} \quad \sum_{i,j,k}\phi_{ijk}(\boldsymbol{x}) = 1. \tag{7}$$

We define $\phi_{ijk}(\boldsymbol{x})$ as a product of one-dimensional interpolation functions $\phi_{ijk}(\boldsymbol{x}) = \phi(\frac{x-x_i}{h})\phi(\frac{y-y_j}{h})\phi(\frac{z-z_k}{h})$, where $h$ is the

voxel grid size, and $\phi$ is a cubic B-spline basis function,

$$\phi(t) = \frac{1}{6}\begin{cases} -3\tau^3 + 3\tau^2 + 3\tau + 1, & |t| \le 1 \\ (1+\tau)^3, & 1 \le |t| \le 2 \\ 0, & \text{otherwise,} \end{cases} \quad (8)$$

where $\tau \equiv 1 - |t|$. Extracting $S$ and $\mathbf{n}$ from $\tilde{B}(\boldsymbol{x})$ defined in this manner provides a smoother iso-surface and normal field (see Figure 3), and greatly reduces sound artifacts.

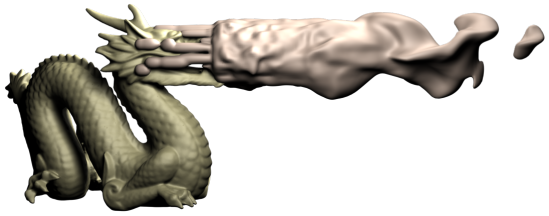


**Figure 3: Flame front surface** *estimated using the method in §3.2.*

### 3.3 Sound Pressure Evaluation

The simulations with which we are trying to synthesize sound are generally only run with time-stepping rates of hundreds of steps per second. To evaluate a pressure signal at audio sampling rates we interpolate the original simulation-rate signal. The pressure signal used to generate sound has the form

$$p(t) = \frac{d}{dt}\int_{S(t)} \mathbf{u}(\boldsymbol{x},t)\cdot\mathbf{n}(\boldsymbol{x},t)\, dS(\boldsymbol{x}) = \frac{dI}{dt}. \quad (9)$$

Since $I(t)$ is known only at discrete time steps $t_0, t_1, t_2, \ldots, t_N = 0, \Delta t, 2\Delta t, \ldots, N\Delta t$, we use interpolation functions $\psi_i(t)$ to form its interpolated signal,

$$\tilde{I}(t) = \sum_{i=0}^{N} \psi_i(t) I(t_i) \quad \text{where} \quad \sum_{i=0}^{N} \psi_i(t) = 1. \quad (10)$$

The sound pressure $p(t)$ can then be evaluated at any point in time,

$$p(t) = \sum_{i=0}^{N} \psi_i'(t) I(t_i). \quad (11)$$

We use a Mitchell-Netravali cubic filter [Mitchell and Netravali 1988] for our interpolation function $\psi_i(t)$.

## 4 Spectral Bandwidth Extension

In this section, we present a bandwidth extension method for adding physics-driven power-law noise to enhance low-frequency fire sounds. Using time-frequency processing we modify our sounds' frequency spectrum so that, beyond a certain frequency, it matches a preferred fire spectra [Zölzer and Amatriain 2002; Marelli et al. 2010]. Recall from §2.3 that experimental and theoretical results suggest that in the mid- to high-frequency range, flame sounds exhibit a power-law spectrum $f^{-\alpha}$, with $\alpha \in [2.2, 3.4]$.

**Background (spectral noise synthesis):** Given a power spectrum $f^{-\alpha}$, the corresponding frequency-domain pressure amplitudes are $|\tilde{p}(f)| = f^{-\alpha/2}$. To synthesize time-domain noise, we first assign uniformly random phases in the range $\theta_j = [-\pi, \pi]$ to each frequency-domain sample $j$ to obtain a frequency-domain sound pressure, $\tilde{p}_j(f) = f_j^{-\alpha/2} e^{-i\theta_j}$. Next, time-domain noise with the desired spectrum is recovered by taking the inverse Fourier transform, $N(t) = IFFT(\tilde{p})$. In practice, to form $N(t)$ we first apply a high pass filter $F_{high}$ to $\tilde{p}$ to suppress frequencies below a cutoff $f_{cut}$. This ensures that the main content of $N(t)$ is indeed high-frequency noise rather than low-frequency rumbling.

**Algorithm 1:** Extends a low-frequency sound with power-law noise

**input** : pressure, $w_H$, $\alpha$, $G_{blur}$
**output**: pressureExtended
**begin**
  pressureExtended $\leftarrow 0$
  **for** $i \leftarrow 0, 1, 2, \ldots$ **do**
    $w_i(n) \leftarrow \begin{cases} w_H - |n - iw_H| & \text{if } |n - iw_H| \le w_H \\ 0 & \text{otherwise} \end{cases}$
    $[p_{W,i}, N_{W,i}] \leftarrow$ `WindowSignal` (pressure,$w_i$,$\alpha$)
    $\beta \leftarrow$ `ChooseBeta` ($p_{W,i}$, $N_{W,i}$, $\alpha$, $G_{blur}$)
    $\tilde{p}_{W,blend} \leftarrow$ `Blend` (`FFT` ($p_{W,i}$), $\beta$ `FFT` ($N_{W,i}$))
    $p_{W,blend} \leftarrow$ `IFFT` ($\tilde{p}_{W,blend}$)
    pressureExtended $\leftarrow$ pressureExtended + $p_{W,blend}$
  **return** pressureExtended

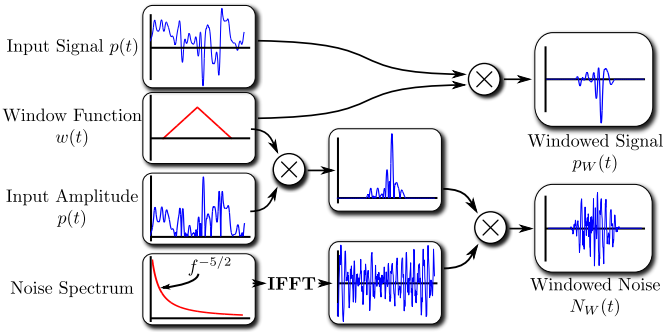
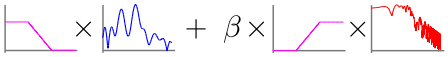



**Figure 4: WindowSignal function from Algorithm 1:** *Given the input signal $p(t)$, we first construct a windowed version $p_W(t)$ by multiplying with a linear window function $w(t)$. A noise signal $N(t)$ is generated by the inverse Fourier transform of a power law spectrum, and scaled by $|p(t)w(t)|$ to produce a windowed noise signal $N_W(t)$.*

**Spectral noise extension:** In order to appropriately synchronize power-law noise with simulated low frequency content, we choose a cutoff frequency $f_{cut}$ at which we wish to begin replacing simulated low-frequency content with synthesized noise. In all of our examples we use the Nyquist frequency of our simulation's 360 Hz time-stepping rate: $f_{cut} = 180$ Hz. Next, we divide our low-frequency pressure signal into windows of half-width $w_H$, where $w_H$ is an integer number of samples. Algorithm 1 describes a time-frequency processing method to introduce high-frequency content into each windowed signal. The windowed input signal $p_W(t)$ and a spectral noise function $N_W(t)$ are computed as shown in Figure 4; we scale the noise by the amplitude of the windowed pressure signal in order to synchronize its temporal structure. We wish to blend together $p_W(t)$ and $N_W(t)$ such that we preserve the original behavior of the pressure spectrum in the neighborhood of $f_{cut}$ and also so that the resulting pressure spectrum does not have any harsh discontinuities. This is done by choosing an appropriate scaling factor $\beta$ for $N_W(t)$. We compute the spectra for each signal, $\tilde{p}_W(f) = \mathbf{FFT}(p_W(t))$ and $\tilde{N}_W(f) = \mathbf{FFT}(N_W(t))$, and linearly blend the two spectra (using high/low pass filters in some range surrounding $f_{cut}$),

$$\tilde{p}_{W,blend}(f;\beta) = F_{low}(f)\, \tilde{p}_W(f) + \beta\, F_{high}(f)\, \tilde{N}_W(f), \quad (12)$$

which can be illustrated as follows:



We estimate $\beta$ by requiring that the integrated power of the blended signal $\tilde{p}_{W,blend}$ match that of the original input signal $\tilde{p}_W$,

$$\int |\tilde{p}_{W,blend}(f;\beta)\, g(f)|^2\, df = \int |\tilde{p}_W(f)\, g(f)|^2\, df, \quad (13)$$

where $g(f)$ is a weighting function used to ensure that signal power is only considered in a blend region near $f_{cut}$. We illustrate (13) as



We solve (13), a quadratic equation, for $\beta > 0$. In our implementation, $g(f)$ is a Gaussian with mean $f_{cut}$ and standard deviation $\sigma_{cut}$. Finally, we use nearly the same parameters for all of our bandwidth-extension examples (see Table 1).

| $\alpha$ | $f_{cut}$ | $\sigma_{cut}$ | $w_H$ | Blend range |
|---|---|---|---|---|
| 2.5–3.5 | 180.0 Hz | 10 Hz | 500 samples (11.34 ms) | [165.0Hz, 195.0Hz] |

**Table 1: Parameters for Spectral Bandwidth Extension:** *Parameters used for all spectral bandwidth extension results. Blend range refers to the linear-blending range used by $F_{low}(f)$ and $F_{high}(f)$.*

# 5 Synchronized Sound Texture Synthesis

The bandwidth extension algorithm presented in §4 adds richness to sounds synthesized with the methods of §3 by adding synchronized and appropriately scaled noise with a physically plausible power spectrum. However, recorded sounds and high-speed flame videos indicate that additional meso-scale temporal structure exists. This structure could be added by synchronizing detailed recordings of fire sound with simulated flames, say, by scaling sound volume according to flame size. However, this simple approach would lack appropriate synchronization, e.g., a large, but steady flame makes very little noise. In this section, we propose a sound texture synthesis algorithm for synthesizing synchronized fire-sound details.

## 5.1 Synthesizing Fire-Sound Details

We synthesize fire-sound details in a coarse-to-fine manner using a modified texture synthesis algorithm most similar to [Wei and Levoy 2000]. Inputs to our algorithm are a fire sound signal $p_S$ synthesized using the methods of §3, and a training audio clip $p_T$, e.g., a recording of real flame sounds. Our audio-detail-transfer method synthesizes a sound clip with the low-frequency temporal structure of $p_S$, but the fiery details of $p_T$.

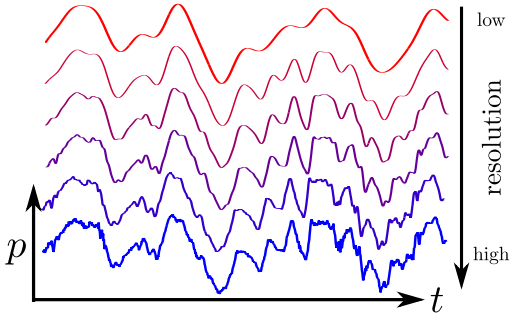



**Figure 5: Fire sound Gaussian pyramid** *for a 23ms training sound $p_T$. All fire-sound details are synthesized in a coarse-to-fine manner using an $L=6$ level pyramid as shown here.*

Analogous to [Wei and Levoy 2000], we construct 1D Gaussian pyramids [Burt and Adelson 1983] $G_T$ and $G_S$ from our training and low-frequency input data, $p_T$ and $p_S$. We refer to $G_T$ and $G_S$ as the "training" and "signal" pyramids, respectively (Figure 5 illustrates $G_T$ for a short fire sound). Let $G_T(\ell)$ and $G_S(\ell)$ refer to the samples stored at level $\ell = 1 \ldots L$, where $L$ is the coarsest level. If $N_{T,\ell}$ (similarly, $N_{S,\ell}$) is the number of samples in level $\ell$ of the training (signal) pyramid, then we write the sequence of samples in the training (signal) pyramid as

$$p^0_{T,\ell}, \ldots, p^{N_{T,\ell}-1}_{T,\ell} \quad \left(p^0_{S,\ell}, \ldots, p^{N_{S,\ell}-1}_{S,\ell}\right). \tag{14}$$

A key distinction between our approach and that of Wei and Levoy [2000] is that we build $G_S$ from our input fire-sound signal $p_S$, rather than initializing it with random noise. We zero the contents of levels $1, \ldots, L-1$, only retaining $p_S$ information in $G_S(L)$. Retaining the low-frequency signal enables us to synthesize synchronized fire-sound details. We synthesize levels of the signal pyramid from coarse to fine; synthesis starts at level $L-1$, since $G_S(L)$ is specified by the input data $p_S$.

## 5.2 Windowed Hierarchical Synthesis

Similar to Wei and Levoy [2000], we synthesize the next "pixel" by finding an approximate nearest neighbor in a dictionary of multi-level neighborhood samples. However, for reasons of temporal coherence and efficiency, we choose to synthesize short, overlapping audio windows instead of individual samples. Consider windows on some level $\ell$ with window centers, $c^i = ih,\ i \in \mathbb{N}$; let the window half-width (in samples) be $h \in \mathbb{N}$ (we use $h=4$) so that window $i$ has sample indices in the range $\Omega^i \equiv \left[c^i - h, c^i + h\right]$. We synthesize level $\ell$ of $G_S$ window-by-window in order of increasing window index, $i$. Suppose that $G_S(\ell)_{i-1}$ is a partially synthesized signal in which windows $0, 1, \ldots, i-1$ have been computed. We form $G_S(\ell)_i$ by adding signal $p_{new}$ to the index range $\Omega^i$, scaled by a linear hat function $W_i[n] = h - \left|n - c^i\right|$.

$p_{new}$ is chosen from a set of training windows produced from $p_T$. Each level $\ell$ of $G_T$ is divided in to windows with index ranges $\Omega^i$. Window $i$ is associated with a feature-value pair $\left(\mathbf{f}^i_{T,\ell}, \mathbf{p}^i_{T,\ell}\right)$ where $\mathbf{p}^i_{T,\ell}$ is the sub-vector of $G_T(\ell)$ with sample indices $\Omega^i$. $\mathbf{f}^i_{T,\ell}$ is a two-level *feature vector* built from samples in $G_T(\ell)$ occurring immediately before window $i$ in the range

$$\left[c^i_\ell - hh_f, c^i_\ell - h\right], \tag{15}$$

and neighboring samples from $G_T(\ell+1)$ in the range

$$\left[(i/2)\,h - hh_f, (i/2)\,h + hh_f\right], \tag{16}$$

where $h_f$ is a window half-width multiplier (we use $h_f = 3$, resulting in features with 46 dimensions). In practice, these samples do not lie at integral indices, in which case we use linear interpolation to determine their values. See figure 6 for an illustration of the structure of a feature vector. We build a dictionary $\mathbb{D}_\ell$ of feature-value pairs for each level $\ell = 1, \ldots, L-1$ (level $L$ is omitted because it is initialized with our low-frequency fire sound). To synthesize window $i$ of level $\ell$ we build a feature vector of samples from $G_S(\ell)$ and $G_S(\ell+1)$ indexed by (15) and (16). $p_{new}$ is chosen by performing a nearest neighbor search in the feature vector space of $\mathbb{D}_\ell$. Figure 6 illustrates the process of building $\mathbb{D}_\ell$ and synthesizing $G_S(\ell)$ for $h = 2, h_F = 2$ (we use $h = 4, h_F = 3$ in practice).

To improve efficiency, we search for approximate rather than exact nearest neighbors. Given a feature $\mathbf{f}^i_{S,\ell}$ whose nearest neighbor is distance $d_{opt}$ away, we find a feature whose distance $d$ from $\mathbf{f}^i_{S,\ell}$ is no more than $(1+\epsilon)\,d_{opt}$. In all examples, we use $\epsilon = 1$. Larger values will improve performance at the cost of sound quality. Approximate nearest neighbor queries are implemented using the *ANN* library (`http://www.cs.umd.edu/~mount/ANN`).

## 5.3 Dynamic Range Matching

The input and training signals $p_S$ and $p_T$ may have vastly different dynamic ranges. This problem must be addressed for the synthesis algorithm in §5.2 to produce meaningful output. For instance, if $p_S$ is much louder than $p_T$ then there may be little temporal coherence from window-to-window in the nearest neighbors sampled from the dictionary $\mathbb{D}_\ell$, resulting in an sound clip with harsh temporal discontinuities. While manual normalization of $p_S$ and $p_T$ can help,

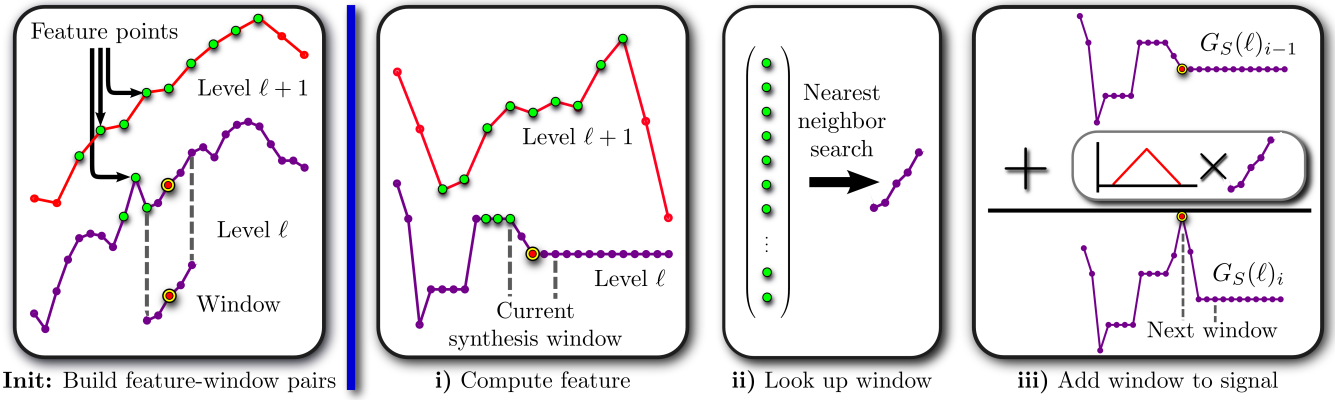



**Figure 6: Sound texture synthesis** ($h = 2, h_F = 2$)**:** *Initialize the synthesis algorithm by building dictionaries $\mathbb{D}_\ell$ of feature-window pairs for levels $\ell = 1, \ldots, L-1$ of the training pyramid. Then, for levels $\ell = L-1, \ldots, 1$ of the signal pyramid; for window $i = 0, 1, 2, \ldots$ in level $\ell$;* **(i)** *Build the window's feature vector,* **(ii)** *Look up the nearest neighbor in the training dictionary $\mathbb{D}_\ell$,* **(iii)** *add the resulting window (scaled by a linear hat function) to the signal at level $\ell$ and proceed to the next window.*

an automated solution to this problem is desirable.

We use a histogram-matching procedure similar to the one proposed in [Heeger and Bergen 1995]. We construct cumulative distribution functions $F_S$ and $F_T$ for the amplitudes of the lowest-resolution signals, $G_S(L)$ and $G_T(L)$, respectively. When synthesizing window $i$ from level $L-1$ of $G_S$ the feature vector $\mathbf{f}^i_{S,L-1}$ includes samples from level $L$ in the range given by (16). We compute the average magnitude of these samples, $p_{abs,avg}$, then evaluate

$$p_{abs,training} = F_T^{-1}\left(F_S\left(p_{abs,avg}\right)\right). \tag{17}$$

Finally, when constructing $\mathbf{f}^i_{S,L-1}$ we scale all samples from $G_S(L)$ by $r = p_{abs,training}/p_{abs,avg}$. Intuitively, if exactly 90% of the signal data has magnitude below $p_{abs,avg}$ then $r$ is chosen so that 90% of the training data lies below $r\ p_{abs,avg}$. Note that this procedure only takes place when choosing feature vector entries from level $L$ of the signal pyramid.

## 6 Results

We now present sound synthesis experiments for a variety of combustion scenarios. Please see the accompanying video for animation and sound rendering results.

**Implementation Details:** All flame simulations were performed using Houdini's Flame solver (based on the *blue core* model described in 2.1) or Houdini's Pyro FX solver (based on the alternate combustion model described in 2.1) and run at a time-stepping rate of 360Hz. Typical simulation times ranged between 2 and 10 hours depending on the length and resolution of the animation and the particular machine used. Iso-surface extraction is carried out using a custom Houdini surface operator (SOP) implemented using the *Houdini Development Kit* (HDK). Surfacing took between 5-20s per time step on a single core. We exploit sample-level parallelism and evaluate surfaces as a post-process in parallel on a cluster. Velocity flux integrals are computed on triangles of $S$ using a one-point quadrature rule. All scenes were rendered using Side Effects Software's *Mantra* renderer. Training audio clips were selected from the *Ultimate Fire* sound library[1]. For texture sound synthesis, all training and input signals $p_T$ and $p_S$ had a 44.1 kHz sampling rate. Prior to either bandwidth extension or sound texture synthesis, we pre-process sound outputs from the synthesis method in §3 using a high-pass filter (with 30Hz cutoff frequency) to suppress low-frequency fluctuations which are inaudible but tend to produce loud, undesirable content when extended via §4 or §5.

[1] http://www.therecordist.com/soundbox-sfx/soundbox-pro/ultimate-fire

**Method Comparison:** We provide comparisons between three sound synthesis approaches:

1. **Low-frequency sound synthesis:** Sounds produced using the velocity flux-based sound model presented in §3.
2. **Noise-based bandwidth extension:** Sounds produced with the bandwidth extension algorithm from §4. We note that although these sounds have a physically plausible power spectrum, they tend to lack interesting temporal behavior beyond what is predicted by the low-frequency synthesis method. Moreover, since all sounds are synthesized using the same noise model, sounds produced using this method tend to be very uniform across a variety of combustion scenarios.
3. **Sound texture synthesis:** Sounds generated using the texture synthesis approach detailed in §5. By varying the choice of training data used in this process, we find that we can produce a richer variety of sounds than those produced by pure noise-based bandwidth extension. Table 2 provides synthesis timings and statistics.

| Example | $p_T$ length (s) | $p_S$ length (s) | $\lvert\mathbb{D}_1\rvert$ | Synthesis time (s) | |
|---|---|---|---|---|---|
| | | | | BE | STS |
| Dragon | 15 | 9 | 167694 | 223 | 56 |
| Candle | 10 | 6 | 112070 | 151 | 86 |
| Torch | 16 | 5 | 179808 | 63 | 41 |
| Flame Jet | 15 | 10 | 167694 | 256 | 54 |
| Burning Brick | 8 | 5 | 98310 | 64 | 20 |

**Table 2: Synthesis statistics** *for bandwidth extension (BE) and sound texture synthesis (STS) examples. $\lvert\mathbb{D}_1\rvert$ is the number of features at the highest resolution level of the training pyramid $G_T$. BE timings are for an unoptimized Matlab implementation.*

**EXAMPLE (Dragon):** We model a fire-breathing dragon roughly 1.2m in height. Fuel inflow from the dragon's mouth is varied to achieve dynamic variation in the sound.

**EXAMPLE (Candle):** This scene models a small candle-like flame blown by a turbulent wind force (see Figure 7 (Left)). Spectral bandwidth extension produces plausible results for this example due to the lack of complex mid-to-high frequency structure in small flames such as this.

**EXAMPLE (Torch):** A burning torch is animated to swing through the domain (see Figure 7 (Middle)) using Houdini's flame solver, which makes use of a blue core model [Nguyen et al. 2002] with detonation shock dynamics [Hong et al. 2007].

**EXAMPLE (Flame Jet):** Fuel is injected with high velocity in to a closed 1.5m wide box (see Figure 7 (Right)). Fuel inflow is an-

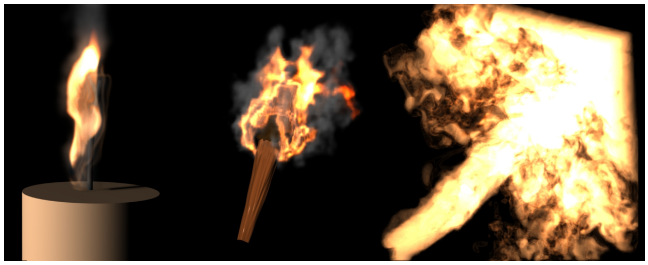


**Figure 7: Candle, Torch & Flame Jet Examples**

imated to turn on and off periodically. As the fuel is shut off, we hear a "burn-off" effect as the remaining fuel is consumed.

**EXAMPLE (Burning Brick):** We model a small rectangular emitter animated to move rapidly through the domain (see Figure 8). As the source is moved, we hear characteristic ruffling noises.

**COMPARISON (Simulated vs. recorded sound):** Sounds and high-speed video (600 FPS) of a small "fire starter" brick were recorded by the authors. The "Burning Brick" example is modeled to resemble this experiment so that comparisons can be made. Results are also generated in which the recorded sound itself is used as training data for the texture synthesis method from §5.

**COMPARISON (Varying power-law exponent, $\alpha$):** We provide a series of results for the "Burning Brick" simulation synthesized using spectral bandwidth extension with varying power law exponents $\alpha$. $\alpha$ is treated as a parameter that can be tuned to change the character of the output sound. $\alpha = 3.0$ seems to produce reasonable results, while $\alpha = 2.5$ and $\alpha = 3.5$ produce too much high and low frequency content, respectively.

**COMPARISON (Synthesizing sounds with different training data):** We present examples of identical simulation scenarios in which we synthesize sounds using texture synthesis with different training audio clips. We see that we can obtain desirable variation in the sounds produced by changing the training data.

**COMPARISON (With and without dynamic range mapping):** We present an example in which texture synthesis has difficulty producing coherent output when the dynamic re-mapping procedure from §5.3 is omitted.

## 7 Conclusion

We presented a method for synthesizing plausible sounds synchronized with physically based simulations of fire. Our hybrid approach produces sounds from simulations time-stepped at low rates, then introduces high-frequency content as a post-process using either spectral bandwidth extension or texture synthesis techniques.

**Limitations and Future Work:** Our results are somewhat restricted by the limited fidelity of signals produced by the methods of §3. They tend to have a somewhat "bursty" character, and lack some of the mid-frequency "whooshing" behavior present in real flames. Recovering this behavior may require more sophisticated modeling of combustion chemistry in the underlying flame solver, which remains an open problem. An efficient computational model of turbulence and its impact on combustion sound is one potential area for future work. Alternately, it may be possible to produce plausible flame sounds without resorting to the costly fluid simulations used in this work. A sound model that is compatible with low-frequency, low-resolution fluid simulation and can be effectively extended using texture synthesis is another area of potential future research.

The dynamic range mapping technique presented in §5.3 makes our texture synthesis more practical, but in some instances the method still has difficulty producing a suitable, temporally coherent output sound. This can occur in cases when the low-frequency input has a very wide dynamic range, while the training data has a small range, e.g., training samples that consist of steady rumbling sounds.

Physically principled spatialization of sounds generated by our method requires further investigation. The low-frequency sound model (§3) describes monopole sound sources distributed on a moving surface. In principle, this model could be used to produce spatialized multi-channel sound. Spatialization is less clear for sounds synthesized with bandwidth extension or texture synthesis.

**Acknowledgements:** We would like to thank the anonymous reviewers for helpful feedback, Karen James for fire wrangling, and Side Effects Software, Inc. for the *Houdini* 3D animation package. This work was supported in part by the National Science Foundation (HCC-0905506), funding and hardware from Intel (ISTC-VC), fellowships from the Alfred P. Sloan Foundation and the John Simon Guggenheim Memorial Foundation, an NSERC Postgraduate Scholarship, and donations from Pixar, Autodesk, and Vision Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

ABUGOV, D. I., AND OBREZKOV, O. I. 1978. Acoustic noise in turbulent flames. *Combustion, Explosion, and Shock Waves 14*, 606–612.

ANNADANA, R., EV, H., SINHA, D., AND FERREIRA, A. 2007. A Novel Audio Post-Processing Toolkit for the Enhancement of Audio Signals Coded at Low Bit Rates. In *AES 123rd Convention, New York, NY, USA, 2007 October 58*.

BRIDSON, R., HOURIHIM, J., AND NORDENSTAM, M. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics 26*, 3 (July).

BURT, P. J., AND ADELSON, E. H. 1983. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics 2*, 4 (Oct.), 217–236.

CARDLE, M., BROOKS, S., BAR-JOSEPH, Z., AND ROBINSON, P. 2003. Sound-by-numbers: motion-driven sound synthesis. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 349–356.

CHRIGHTON, D. G., DOWLING, A. P., WILLIAMS, J. E. F., HECKL, M., AND LEPPINGTON, F. G. 1992. *Modern Methods in Analytical Acoustics*. Springer-Verlag.

CLAVIN, P., AND SIGGIA, E. D. 1991. Turbulent premixed flames and sound generation. *Combustion Science and Technology 78*, 147–155.

COOK, P. R. 2002. Sound Production and Modeling. *IEEE Computer Graphics & Applications 22*, 4 (July/Aug.), 23–27.

DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. 2003. Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics. *ACM Transactions on Graphics 22*, 3 (July), 539–545.

DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. 2004. Synthesizing sound from turbulent fields using sound textures for interactive fluid simulation. *Computer Graphics Forum 23*, 3 (Sept.), 736–744.

DUBNOV, S., BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., AND WERMAN, M. 2002. Synthesizing sound textures through

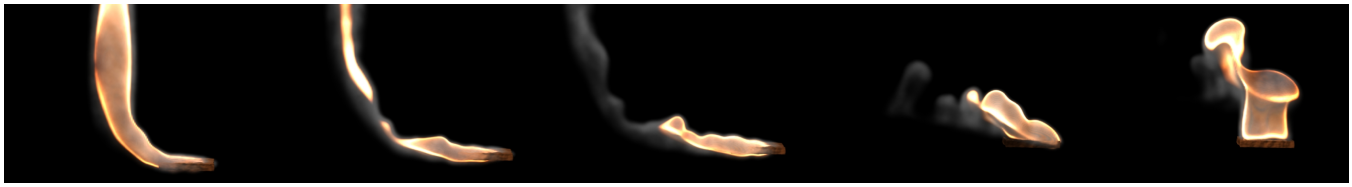


**Figure 8: Burning brick example** *produces characteristic ruffling flame sounds as it moves side to side.*


wavelet tree learning. *Computer Graphics and Applications, IEEE 22*, 4, 38–48.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 341–346.

EFROS, A., AND LEUNG, T. 1999. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, IEEE, 1033–1038.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND ARIKAN, O. 2003. Animating suspended particle explosions. In *Proceedings of SIGGRAPH 2003*, 708–715.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *Proc. of SIGGRAPH 1997*, 181–188.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proc. of SIGGRAPH 1995*, 229–238.

HONG, J.-M., SHINAR, T., AND FEDKIW, R. 2007. Wrinkled flames and cellular patterns. *ACM Transactions on Graphics 26*, 3 (July), 47:1–47:6.

HORVATH, C., AND GEIGER, W. 2009. Directable, High-Resolution Simulation of Fire on the GPU. *ACM Transactions on Graphics 28*, 3 (July), 41:1–41:8.

HOWE, M. S. 2003. *Theory of Vortex Sound*. Cambridge University Press.

IHME, M., PITSCH, H., AND BODONY, D. 2009. Radiation of noise in turbulent non-premixed flames. In *Proceedings of the Combustion Institute*, vol. 32, 1545–1553.

KIM, T., THÜREY, N., JAMES, D., AND GROSS, M. 2008. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics 27*, 3 (Aug.), 50:1–50:6.

LARSEN, E., AND AARTS, R. 2004. *Audio bandwidth extension: application of psychoacoustics, signal processing and loudspeaker design*. Wiley.

LIU, C., LEE, W., AND HSU, H. 2003. High frequency reconstruction for band-limited audio signals. *Proc. DAFX-03, Sept*.

LORENSON, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH 1987*, 163–169.

MARELLI, D., ARAMAKI, M., KRONLAND-MARTINET, R., AND VERRON, C. 2010. Time-frequency synthesis of noisy sounds with narrow spectral components. *IEEE Transactions on Audio, Speech and Language Processing 18*, 8 (Nov.), 399–414.

MCDERMOTT, J., OXENHAM, A., AND SIMONCELLI, E. 2009. Sound texture synthesis via filter statistics. In *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA'09. IEEE Workshop on*, IEEE, 297–300.

MITCHELL, D. P., AND NETRAVALI, A. N. 1988. Reconstruction filters in computer-graphics. In *Proceedings of SIGGRAPH 1988*, 221–228.

MOSS, W., YEH, H., MO HONG, J., LIN, M. C., AND MANOCHA, D. 2010. Sounding liquids: Automatic sound synthesis from fluid simulation. *ACM Transactions on Graphics 29*, 3 (June), 21:1–21:13.

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

O'BRIEN, J. F., COOK, P. R., AND ESSL, G. 2001. Synthesizing sounds from physically based motion. In *Proceedings of SIGGRAPH 2001*, 529–536.

O'BRIEN, J. F., SHEN, C., AND GATCHALIAN, C. M. 2002. Synthesizing sounds from rigid-body simulations. In *ACM SIGGRAPH Symposium on Computer Animation (SCA)*, 175–181.

RAJARAM, R., AND LIEUWEN, T. 2009. Acoustic radiation from turbulent premixed flames. *Journal of Fluid Mechanics 637*, 357–385.

ROADS, C. 2004. *Microsound*. The MIT Press.

SCHWARZ, A., AND JANICKA, J., Eds. 2009. *Combustion Noise*. Springer.

SERRA, X., AND SMITH III, J. 1990. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Comp. Music Journal 14*, 4, 12–24.

STRAHLE, W. C. 1972. Some results in combustion generated noise. *Journal of Sound and Vibration 43*, 1, 113–125.

STROBL, G., ECKEL, G., ROCCHESSO, D., AND LE GRAZIE, S. 2006. Sound texture modeling: A survey. In *Proceedings of the 2006 Sound and Music Computing (SMC) International Conference*, 61–5.

VAN DEN DOEL, K., KRY, P. G., AND PAI, D. K. 2001. Foleyautomatic: Physically based sound effects for interactive simulation and animation. In *Proceedings of SIGGRAPH 2001*, 537–544.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH 2000*, 479–488.

ZHENG, C., AND JAMES, D. L. 2009. Harmonic fluids. *ACM Transactions on Graphics 28*, 3 (Aug.).

ZHENG, C., AND JAMES, D. L. 2010. Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics 29*, 3 (July).

ZÖLZER, U., AND AMATRIAIN, X. 2002. *DAFX: Digital Audio Effects*. John Wiley & Sons Inc.