# Wavelet Turbulence for Fluid Simulation

Theodore Kim, Cornell University
Nils Thürey, ETH Zurich
Doug James, Cornell University
Markus Gross, ETH Zurich

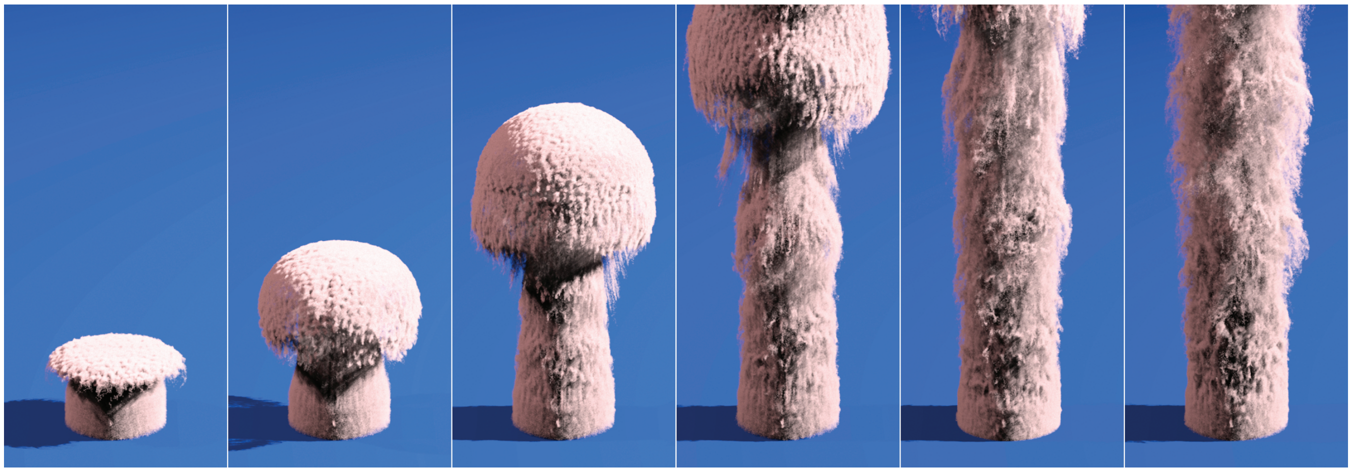

**Figure 1:** *A* $50 \times 100 \times 50$ *smoke simulation with eight turbulence bands added to synthesize an effective resolution of* $12800 \times 25600 \times 12800$. *Post-processing averaged 170s a frame on an 8 core machine and needed 180 MB of extra memory. Particles were used to track density.*

## Abstract

We present a novel wavelet method for the simulation of fluids at high spatial resolution. The algorithm enables large- and small-scale detail to be edited separately, allowing high-resolution detail to be added as a post-processing step. Instead of solving the Navier-Stokes equations over a highly refined mesh, we use the wavelet decomposition of a low-resolution simulation to determine the location and energy characteristics of missing high-frequency components. We then synthesize these missing components using a novel incompressible turbulence function, and provide a method to maintain the temporal coherence of the resulting structures. There is no linear system to solve, so the method parallelizes trivially and requires only a few auxiliary arrays. The method guarantees that the new frequencies will not interfere with existing frequencies, allowing animators to set up a low resolution simulation quickly and later add details without changing the overall fluid motion.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modelling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation;

**Keywords:** turbulence, wavelets, noise, fluids, simulation control

## 1 Introduction

Considerable progress has been made in the last decade in the visual simulation of fluids, but scalability and user interaction still remain significant problems. Most recent methods directly solve the Navier-Stokes or incompressible Euler equations over a mesh. However, if features smaller than a mesh element are required, as is common when simulating large-scale phenomena such as explosions or volcanic eruptions, the mesh must be refined in some way. This results in a linear increase in memory use and a greater than linear increase in the running time.

We instead propose an algorithm that generates small-scale fluid detail procedurally. We use a wavelet decomposition to detect where small-scale detail is being lost, and apply a novel incompressible turbulence function to reintroduce these details. Instead of enforcing the Navier-Stokes equations over all spatial scales, we enforce Navier-Stokes over low frequencies and Kolmogorov's turbulence spectrum over high frequencies. We have eliminated the global dependencies of a linear solver over the grid, so the high-resolution portion of the algorithm parallelizes trivially. The derivation of our method additionally provides insight into the popular Perlin `turbulence()` function [Perlin 1985] and suggests a physical basis for its success. The algorithm requires only the velocity field of an existing fluid simulation as input, and can be run as a fully decoupled post-processing step. This allows users to rapidly iterate on fluid simulation designs until the desired overall behavior is achieved, then later add small-scale details prior to rendering. The fluid solvers commonly used in graphics do not allow this. Increasing the resolution changes the effective viscosity of the fluid, produces significantly different motions at a higher resolution, and invalidates much of the design work done on lower resolutions. In contrast, our method introduces new energies in a band-limited manner that guarantees existing structures are preserved.

Our contributions are as follows:

- An incompressible turbulence function that can generate arbitrary energy spectra.
- A method for estimating the small-scale turbulence that is lost by a simulation, and re-synthesizing it in a way consistent with Kolmogorov theory.
- A method of preserving the temporal coherence of the synthesized turbulence.
- A large- and small-scale fluid detail decoupling that allows the latter to be edited independently.

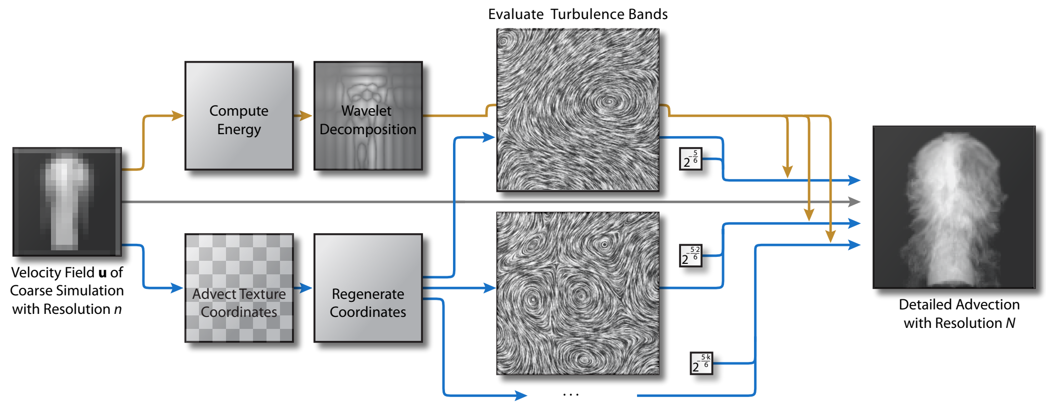
Evaluate Turbulence Bands
Compute Energy
Wavelet Decomposition
Velocity Field **u** of Coarse Simulation with Resolution *n*
Advect Texture Coordinates
Regenerate Coordinates
$2^{-\frac{5}{6}}$
$2^{-\frac{5\cdot 2}{6}}$
$2^{-\frac{5\cdot k}{6}}$
...
Detailed Advection with Resolution *N*

**Figure 2: Overview:** *A low-resolution $n^3$ velocity field* **u** *is used to synthesize a high-resolution $N^3$ density field. Procedural turbulence is added according to the wavelet decomposition of the energy, and the resulting eddies are advected via texture coordinates until they scatter.*

## 2 Previous Works

Jos Stam [1999] introduced to computer graphics the combination of implicit Poisson solvers and semi-Lagrangian advection that is widely used to visually simulate fluids today. Many subsequent works have refined this initial algorithm. Vorticity confinement [Fedkiw et al. 2001], vortex particles [Selle et al. 2005], Back and Forth Error Compensation and Correction (BFECC) [Kim et al. 2005], and MacCormack [Selle et al. 2008] methods all attempt to maximize the detail resolved on a grid by suppressing dissipation. Recent variational methods [Elcott et al. 2007; Batty et al. 2007] have designed discretizations that pursue the same goal. In the best case, these methods allow fluid features that are near the Nyquist limit to be robustly resolved. We consider this “dissipation suppression” problem to be orthogonal to our own, because we are concerned with efficiently resolving frequencies *greater* than the Nyquist limit. We consider this a separate “frequency extension” problem. Consequently, any of the above methods could be used to complement our method.

The simplest method of capturing subgrid fluid detail is uniform refinement. However, this scales memory usage by a factor of eight and increases the running time of the Poisson solver. Adaptive methods [Losasso et al. 2004; Klingner et al. 2006] address this problem, but are not a panacea. Adaptive data structures introduce more memory overhead, and increase implementation and runtime complexity. We instead propose a *synthetic turbulence* model like the ones used in Large Eddy Simulation to capture subgrid details [Scotti and Meneveau 1999; Basu et al. 2004]. These prior methods work to preserve the statistical properties of the sub-grid fluid, but tend to introduce dyadic visual artifacts which our method avoids.

Our method is most similar to the 4D Kolmogorov method in Stam and Fiume [1993] and the procedural method of Bridson et al. [2007]. Stam and Fiume [1993] uses a 4D FFT to generate “ambient turbulence” that is used to add new eddies. While Kolmogorov theory is primarily intended for fully developed, homogeneous turbulence, it provides acceptable visual results. However, small eddies tend to form without a larger precursor, and do not advect along the flow, making the static uniformity of the turbulence apparent. Our more flexible approach generates eddies only where they would physically form and advects the results. Our approach is also entirely local, so, unlike the FFT, it parallelizes trivially. The approach of [Bridson et al. 2007] computes the curl of a user-input potential to produce a velocity field with eddies proportional to the finite difference step size. Our method is complementary, as it could be used to then add smaller eddies in the same way as with any other low-resolution flow.

Rasmussen et al. [2003] used the method of Stam and Fiume [1993] to break up artifacts when emulating a full 3D simulation using 2D slices, and Lamorlette and Foster [2002] used it to add detail to a procedural flame modeling system. Neyret [2003] used the same multi-scale intuition as Kolmogorov to animate textures. Our technique is related, but because we are advecting eddies specifically, we can use a more physically based method to regenerate the texture coordinates.

Recent work on fluid control has focused on keyframe-driven animation [McNamara et al. 2004; Fattal and Lischinski 2004] and motion filaments [Angelidis et al. 2006] that control the low-frequency flow. Our approach complements such control methods because it supports editing of high-frequency components without disturbing low-frequency flow.

## 3 Procedural Wavelet Turbulence

In this section we describe how to efficiently construct an incompressible turbulence function. We adopt the following notation for the remainder of this paper. Bold denotes a vector, and non-bold denotes a scalar. The special variable $\mathbf{x}$ denotes a spatial position, $k$ denotes a spectral band, and $\mathbf{u}$ denotes a velocity field. A carat denotes a wavelet transform, so $\hat{\mathbf{u}}(\mathbf{x},k)$ denotes the spectral component of velocity field $\mathbf{u}$ at position $\mathbf{x}$ in spectral band $k$. Additionally, $n$ always refers to the grid resolution, $n^3$, of $\mathbf{u}$, and $\mathbf{v_x}$, $\mathbf{v_y}$, $\mathbf{v_z}$ refer to the Cartesian unit vectors, ie $\mathbf{v_x} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$.

**Incompressible, Band-Limited Noise:** Wavelet Noise [Cook and DeRose 2005] was developed as a replacement for Perlin Noise [Perlin 1985]. The noise is guaranteed to exist only over a narrow spectral band, which makes more sophisticated filtering possible. We will use this band-limited property as a first step in constructing an incompressible turbulence function.

The Wavelet Noise function $\omega$ is a scalar function, whereas we are interested in vector fields. As was recently demonstrated for graphics [Bridson et al. 2007], a calculus identity can be used to construct a divergence-free vector field by taking the curl of a scalar field. We can respectively construct 2D and 3D vector fields using $\omega$:

$$\mathbf{w}_{2D}(\mathbf{x}) = \left(\frac{\partial\omega}{\partial y}, -\frac{\partial\omega}{\partial x}\right) \quad (1)$$

$$\mathbf{w}(\mathbf{x}) = \left(\frac{\partial\omega_1}{\partial y} - \frac{\partial\omega_2}{\partial z}, \frac{\partial\omega_3}{\partial z} - \frac{\partial\omega_1}{\partial x}, \frac{\partial\omega_2}{\partial x} - \frac{\partial\omega_3}{\partial y}\right) \quad (2)$$

We primarily use the 3D case (Eqn. 2) unless otherwise stated. The 3D case requires three different noise tiles, which we have denoted $\omega_1, \omega_2$ and $\omega_3$, but in practice we use offsets into the same noise tile. The derivatives can be evaluated directly because $\omega$ uses B-spline interpolation. Instead of the usual quadratic B-spline weights, $\left[\frac{t^2}{2}, \frac{1}{2}+t(1-t), \frac{(1-t)^2}{2}\right]$, we apply the derivative weights, $[-t, 2t-1, 1-t]$, in the desired directions. The resultant vector field is guaranteed to be incompressible. The field retains the same band-limited properties of the original signal because differentiation is a linear high-pass filter in the frequency domain, which does not add new frequencies by definition. This is sufficient to ensure that the 2D vector field is band limited. Additionally, we observe that adding two signals together is also linear, so the 3D case is also band limited.

From a visual perspective, $\mathbf{w}$ generates a vector field of randomly distributed, tightly packed eddies of fixed size. Two bands of the function can be seen in Figure 2. Next, we will use $\mathbf{w}$ to generate a vector field according to Kolmogorov's theory of turbulence.

**Kolmogorov Wavelet Turbulence:** Kolmogorov famously showed that, for a homogeneous inviscid fluid, while the local structure of its velocity field may be perpetually chaotic, the global energy spectrum approaches an equilibrium state that can be described in very simple terms [Frisch 1995]. The energy density $e$ of some grid cell $\mathbf{x}$ is its kinetic energy,

$$e(\mathbf{x}) = \frac{1}{2}|\mathbf{u}(\mathbf{x})|^2. \tag{3}$$

The total energy $e_t$ of a grid is then the sum over all grid cells. Kolmogorov's theory deals with the frequency spectrum of $e_t$. While the original theory used the Fourier transform, we use a wavelet transform because it provides both spatial and frequency information. If we compute $e_t$ for each band $k$ of $\hat{\mathbf{u}}(\mathbf{x}, k)$ we obtain an energy spectrum $e_t(k)$. One of the key results of Kolmogorov theory is that the energy spectrum of a turbulent fluid approaches a five-thirds power distribution:

$$e_t(k) = C\varepsilon^{\frac{2}{3}} k^{-\frac{5}{3}}. \tag{4}$$

Where $C$ and $\varepsilon$ are the Kolmogorov constant and the mean energy dissipation rate per unit mass. The $-\frac{5}{3}$ scaling exponent holds for both Fourier and wavelet spectra [Perrier et al. 1995], and the substitution is common in fluid dynamics [Farge et al. 1996].

Using our noise function $\mathbf{w}$, we can procedurally construct a velocity field that produces this power distribution. We first observe that (4) can be rewritten as the recurrence relation:

$$e_t(2k) = e_t(k)2^{-\frac{5}{3}}, \quad e_t(1) = C\varepsilon^{\frac{2}{3}}.$$

The velocities can then be stated analogously using (3):

$$|\hat{\mathbf{u}}(\mathbf{x}, 2k)| = |\hat{\mathbf{u}}(\mathbf{x}, k)|2^{-\frac{5}{6}}, \quad |\hat{\mathbf{u}}(\mathbf{x}, 1)| = 2^{\frac{1}{2}} C^{\frac{1}{2}} \varepsilon^{\frac{2}{6}}. \tag{5}$$

Since $\mathbf{w}(\mathbf{x})$ is band limited, it can be substituted in for $\hat{\mathbf{u}}(\mathbf{x}, k)$. Our final wavelet turbulence function is then a series version of (5):

$$\boxed{\mathbf{y}(\mathbf{x}) = \sum_{i=i_{\min}}^{i_{\max}} \mathbf{w}(2^i \mathbf{x}) 2^{-\frac{5}{6}(i-i_{\min})}.} \tag{6}$$

The variables $[i_{\min}, i_{\max}]$ can be used to control the spectral bands that $\mathbf{y}(\mathbf{x})$ applies to.

**Discussion:** Eqn. 6 shares much of the appeal of Perlin's widely used `turbulence()` function [Perlin 1985]. Perlin's `turbulence()` also sums successive bands of a noise function $p(\mathbf{x})$ to obtain a "visually turbulent" scalar function. It can be stated in terms very similar to ours:

$$\mathtt{turbulence}(\mathbf{x}) = \sum_{i=i_{\min}}^{i_{\max}} p(2^i \mathbf{x}) \frac{1}{2}^{i-i_{\min}}. \tag{7}$$

Our function is essentially a *vector* version of (7) that is also band-limited, guarantees incompressibility, and produces the Kolmogorov power distribution. Interestingly, $2^{-\frac{5}{6}} \approx 0.56$, which is close to the heuristic $\frac{1}{2}$ value used by Perlin. If scalar wavelet noise $w$ were used instead of $p$, and $2^{-\frac{5}{6}}$ instead of $\frac{1}{2}$, a Kolmogorov-Obukhov-Corrsin *scalar turbulence spectrum* would be obtained [Shraiman and Siggia 2000], which suggests a physical reason for Perlin's success at generating visually turbulent textures.

## 4 High-Resolution Fluid Synthesis

We now show how to use the turbulence function $\mathbf{y}(\mathbf{x})$ to add new high-frequency components to $\mathbf{u}$. First, we motivate our approach by discussing the intuition that underlies Kolmogorov's theory.

**Background:** Physically, the five-thirds distribution occurs because of *scattering* [Frisch 1995]. As an eddy is advected by an incompressible field, it is stretched in one direction and compressed in another. Eventually these deformations break the eddy into two eddies of half the size. This process is called *forward scattering*. The opposite phenomena, *back scattering*, occurs when smaller eddies combine to form larger ones, but forward scattering usually dominates. Kolmogorov's five-thirds spectrum describes the energy distribution after sufficient time has passed that eddies injected at a fixed scale (the *integral scale*) have forward scattered into higher frequencies. At much higher frequencies, viscosity becomes dominant, so at a second scale (the *ultraviolet cutoff*) the energies start to dissipate at a much faster rate than the five-thirds exponent. In graphics, dissipation damps out interesting fluid detail, so the viscous term is usually dropped from the Navier-Stokes equations. Theoretically this places the ultraviolet cutoff at the Nyquist limit, $\frac{n}{2}$ for an $n^3$ mesh. As mentioned in §2, significant dissipation still occurs before the Nyquist limit, and many techniques have been developed to address this issue. We instead focus on the separate problem of placing the ultraviolet cutoff *beyond* the Nyquist limit.

**Injecting Turbulence:** Our goal is to synthesize a high-resolution $N^3$ density field $D$ from a low-resolution $n^3$ velocity field $\mathbf{u}$. We use lowercase to denote variables at the lower $n^3$ resolution and uppercase for fields at the higher $N^3$ resolution. We define an interpolation function $I(\mathbf{u}, \mathbf{X})$ that interpolates $\mathbf{u}$ at the high-resolution location $\mathbf{X}$, and $\mathcal{A}(\mathbf{U}, D)$ as the advection of $D$ by $\mathbf{U}$.

We will now focus on synthesizing an $N^3$ velocity field $\mathbf{U}$. The simplest method is interpolation: $\mathbf{U}(\mathbf{X}) = I(\mathbf{u}, \mathbf{X})$. This smooths out the velocities according to the interpolation method, but it does not generate any new eddies in the new $[n, \frac{N}{2}]$ spectral bands. A more sophisticated method is to compute the energy $e_t(\frac{n}{2})$ of the smallest eddies in $\mathbf{u}$, and use it to weight our turbulence function:

$$\mathbf{U}(\mathbf{X}) = I(\mathbf{u}, \mathbf{X}) + 2^{-\frac{5}{6}} \cdot e_t\left(\frac{n}{2}\right) \cdot \mathbf{y}(\mathbf{X}) \tag{8}$$

We only want to inject energy into the new $[n, \frac{N}{2}]$ bands, so we set $i_{\min} = \log_2 n$ and $i_{\max} = \log_2 \frac{N}{2}$ in $\mathbf{y}(\mathbf{X})$. Intuitively, this approach assumes that the energy spectrum of $\mathbf{u}$ follows the Kolmogorov distribution, computes the last resolved value, $e_t(\frac{n}{2})$, and uses it to extrapolate energies over the new $[n, \frac{N}{2}]$ bands.

This method approximates the high-frequency components of $\mathbf{U}$ as fully developed, homogeneous turbulence in a manner similar to Stam and Fiume [1993]. While this can produce acceptable results, because $\mathbf{y}(\mathbf{X})$ is weighted globally, it can spontaneously produce small-scale eddies in regions where there is no larger-scale precursor. The weighting should instead vary spatially, with turbulence added only when an eddy in $\mathbf{u}$ forward scatters into a previously unresolvable frequency. The resulting turbulence then needs to be advected along with the flow.

**Detecting Scattering:** Forward scattering can be detected by weighting $\mathbf{y}(\mathbf{X})$ by $\widehat{e}(\mathbf{x}, \frac{n}{2})$ instead of $e_t(\frac{n}{2})$. This locally extrapolates the energy spectrum using the same intuition as Eqn. 8. Like Neyret [2003], we then advect a set of texture coordinates $\mathbf{c} = (c_u, c_v, c_w)$ along with the flow, and evaluate $\mathbf{y}$ using the advected value. If we detect that the local texture coordinates are causing $\mathbf{y}$ to deviate too far outside of its original spectral band, they are regenerated to the original local values. The method in Neyret [2003] uses a heuristic strain criterion to trigger regeneration because it is intended for generic texture advection. We are specifically advecting eddies, so we can use more physically based criteria. We quantify the amount of local deformation using the Jacobian of the texture coordinates, which we denote $\mathbf{J}(\mathbf{c}(\mathbf{x}))$.

For the eigenvalues $\lambda(\mathbf{c}(\mathbf{x}))$ of $\mathbf{J}(\mathbf{c}(\mathbf{x}))$, if $\max(|\lambda(\mathbf{c}(\mathbf{x}))|) \geq 2$, from the physical standpoint, the local eddy has forward scattered into a higher spectral band. If $\min(|\lambda(\mathbf{c}(\mathbf{x}))|) \leq \frac{1}{2}$, then the eddy has back scattered to a lower band. In both cases, the eddy is no longer physically (or visually) coherent, so the texture coordinate should be regenerated.

**Texture Distortion:** As the advected texture coordinates stretch and rotate, $\mathbf{y}$ does as well. This potentially violates incompressibility because, as the texture coordinates deform, the derivatives $[\partial\omega/\partial c_u,\ \partial\omega/\partial c_v,\ \partial\omega/\partial c_w]$ no longer correspond to derivatives along the Cartesian axes, $[\partial\omega/\partial x,\ \partial\omega/\partial y,\ \partial\omega/\partial z]$. The derivatives can be obtained by projecting the Cartesian axes into texture space and taking the directional derivative:

$$\left[\frac{\partial\omega}{\partial c_u}\frac{\partial\omega}{\partial c_v}\frac{\partial\omega}{\partial c_w}\right]\mathbf{J}(\mathbf{c}(\mathbf{x}))^{-1}\left[\begin{array}{ccc}\mathbf{v_x} & \mathbf{v_y} & \mathbf{v_z}\end{array}\right] = \left[\frac{\partial\omega}{\partial x}\frac{\partial\omega}{\partial y}\frac{\partial\omega}{\partial z}\right]$$

The resulting Cartesian derivatives are then used in (2). We denote a modified turbulence function that takes in a texture coordinate and performs this projection as $\mathbf{z}(\mathbf{c})$.

**Final Algorithm:** Our final equation for generating a high-resolution velocity field is:

$$\mathbf{U}(\mathbf{X}) = I(\mathbf{u}, \mathbf{X}) + 2^{-\frac{5}{6}}\, I\left(\widehat{e}\left(\mathbf{x}, \frac{n}{2}\right), \mathbf{X}\right)\mathbf{z}(I(\mathbf{c}, \mathbf{X})). \qquad (9)$$

As before, the values of $i_{\min} = \log_2 n$ and $i_{\max} = \log_2 \frac{N}{2}$ are used to evaluate $\mathbf{z}$ over the appropriate spectral bands. The full algorithm is described below, and illustrated in Figure 2:

```
SYNTHESIZE-FLUID($\mathbf{u}$)
1  $\mathcal{A}(\mathbf{u},\mathbf{c})$
2  Compute $\widehat{e}(\mathbf{x}, \frac{n}{2})$, $\mathbf{J}(\mathbf{c}(\mathbf{x}))$, $\lambda = \lambda(\mathbf{c}(\mathbf{x}))$
3  if $\max(|\lambda|) \geq 2$ or $\min(|\lambda|) \leq \frac{1}{2}$
4     then Regenerate $\mathbf{c}(\mathbf{x})$.
5  Synthesize $\mathbf{U}$ using (9)
6  $\mathcal{A}(\mathbf{U}, D)$
7  return D
```

**Complexity:** Almost all the steps occur on the smaller $n^3$ grid, with the $N^3$ grid only being used for the final generation of $\mathbf{U}$ and advection of $D$. The algorithm requires six additional arrays of size $n^3$: $\mathbf{c}$, $\min(|\lambda|)$, $\max(|\lambda|)$ and $\widehat{e}$. The values of $\mathbf{J}$ can be discarded once $\lambda$ is computed. The large $N^3$ arrays for $\mathbf{U}$ and $D$ are a drawback because they increase memory use to $O(N^3)$. However, if $\mathcal{A}(\mathbf{U}, D)$ is implemented using a semi-Lagrangian scheme, only $D$ is instantiated explicitly because each $\mathbf{U}(\mathbf{X})$ is discarded after computing $D(\mathbf{X})$. If particles are used to track the density, even the explicit $D$ is unnecessary and $O(n^3)$ memory use is achieved.

**Obstacles and Control:** Obstacles can be incorporated with minor modifications. The main issue is that, if velocities inside an obstacle are set to zero, the discontinuity in $\mathbf{u}$ can cause a jump in $\widehat{e}(\mathbf{x}, \frac{n}{2})$. To prevent this we extrapolate energy values from the obstacle boundary inwards using a fast marching method prior to computing $\widehat{e}(\mathbf{x}, \frac{n}{2})$. In order to give a user control over the turbulence, $\mathbf{y}$ can be weighted by an arbitrary volumetric function $v(\mathbf{X})$ in addition to $\widehat{e}(\mathbf{x}, \frac{n}{2})$. A user can amplify turbulence in regions by setting $v(\mathbf{x}) > 1$ or can suppress detail by specifying $v(\mathbf{x}) < 1$. In either case, the initial $\widehat{e}(\mathbf{x}, \frac{n}{2})$ value provides a good default setting.

## 5 Results

The $I$ method in §4 is ideally the wavelet upsampling method. However, we compared the results of the upsampler with simple linear interpolation and found the difference negligible, and so favored the slightly faster linear interpolation. The $\mathcal{A}$ method we used was the MacCormack method of Selle et al. [2008]. We used a standard fluid solver [Fedkiw et al. 2001] for our low- resolution simulations, but added a small amount of heat diffusion to stimulate velocities outside of existing low-resolution densities. We used co-located grids to minimize the number of times (9) is evaluated. On a staggered grid, cell-centered averages could be computed and then sent to (9) to achieve similar savings.

Figure 1 shows a particle simulation of smoke, and Figure 5 shows a close-up of the same simulation with an obstacle. We are able to achieve a very high effective resolution because a high-resolution grid is unnecessary. In Figures 3 and 6, we show a simulations of smoke interacting with static obstacles. In Figure 4, the control possible with our method by activating alternating vertical slabs of turbulence can be seen. Complete descriptions are available in the figure captions. Comparisons of our algorithm to Stam and Fiume [1993], Bridson et al. [2007], and an explicit full-resolution simulation are available in the enclosed video. For the explicit comparison, we downsampled an existing high-resolution simulation and then re-synthesized the discarded bands. Our algorithm appears to resolve more high-frequency detail, as it does not suffer from dissipation near the Nyquist limit. Preliminary timings suggest our method is significantly faster than an equivalent full-resolution simulation. For a $250^3$ simulation, the single-threaded version of our method ran roughly seven times faster than the full solver.

All the steps of the complete algorithm only require local support, so the computation parallelizes trivially. We successfully achieved significant speedups by adding a single OpenMP directive to the outermost loop. The main computation, the evaluation of $\mathbf{z}$, runs 3.7 times faster on a four core workstation. This also suggests that the algorithm will perform very well on GPUs.

## 6 Conclusions

We have demonstrated a wavelet method that is suitable for adding detail to existing fluid simulations as a user-controlled post-process. While we have demonstrated results using Eulerian grids, all our method requires is the ability to point-sample a velocity field, so it applies to Lagrangian simulations as well.

The method has several limitations. By design, it does not reproduce the results of explicit high-resolution simulations. This is partly because back scatter from higher bands into the low-resolution simulation has been explicitly suppressed. A physically based method of estimating back scatter could be developed to address this. Also, the quality of obstacle interaction is totally dependent on the interaction quality at low-resolution. As low-resolution results improve, our results will improve correspondingly. It is also possible for the wavelet transform to introduce small coefficients into the high-frequency bands, resulting in non-zero turbulence breaking up an otherwise laminar flow. While clamping is a coarse solution, the issue is worth further study.

Several issues were found to be visually unimportant. First, the coordinates $\mathbf{c}$ should be regenerated on a per-band basis, as higher frequency bands deform faster. Per-band coordinates were implemented and did not make an appreciable visual difference, so they were discarded for simplicity. The more relevant guarantee that the bands of $\mathbf{y}$ do not impinge on those of $\mathbf{u}$ remains intact. Second, a direct visualization of $\mathbf{c}$ shows significant popping because we do not perform any blending during regeneration. However, the popping does not introduce noticeable artifacts in the vector fields because it only occurs after the eddy is no longer visually relevant. Finally, the non-uniform weighting of $\mathbf{z}$ by $\widehat{e}(\mathbf{x}, \frac{n}{2})$ introduces some compressibility. The solution is to project all the energies into texture space and weight the noise tile by $\widehat{e}(\mathbf{x}, \frac{n}{2})$. We found these extra computations unnecessary, as the error introduced is small compared to the error of the advection method.

The per-band weights of our method can be modified arbitrarily, allowing for 'spectral shaping' in the same way as Perlin noise, and allows many of the procedural texturing tricks used for scalar textures to be applied to the vector regime. The Kolmogorov theory does not account for chemical effects such as those in explosions, but since these effects primarily introduce perturbations to the energy spectrum, they could potentially be captured through shaping. Automatic methods of coupling the weights to existing methods [McNamara et al. 2004] also have the potential to generate very specific, fine-grained fluid detail. Finally, the possibility of a similar procedural method that simulates turbulence on a liquid free surface remains to be investigated.

**Acknowledgements:** The authors would like to thank the SIGGRAPH reviewers for their comments and suggestions; Bernd Bickel, Sebastian Martin, Miguel Otaduy, and Filip Sadlo for helpful discussions; support from the National Institutes of Health, NSF CAREER (CCF-0652597), Alfred P. Sloan Foundation Research Fellowship, and donations from Intel, Pixar, Autodesk and NVIDIA.

## References

ANGELIDIS, A., NEYRET, F., SINGH, K., AND NOWROUZEZAHRAI, D. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *ACM SIGGRAPH/EG Symposium on Computer Animation (SCA)*.

BASU, S., FOUFOULA-GEORGIOU, E., AND PORTE-AGEL, F. 2004. Synthetic turbulence, fractal interpolation, and large-eddy simulation. *Physical Review E*, 026310.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. In *Proceedings of ACM SIGGRAPH*.

BRIDSON, R., HOURIHAN, J., AND NORDENSTAM, M. 2007. Curl-noise for procedural fluid flow. In *Proceedings of ACM SIGGRAPH*.

COOK, R., AND DEROSE, T. 2005. Wavelet noise. In *Proceedings of ACM SIGGRAPH*.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Transactions on Graphics*.

FARGE, M., KEVLAHAN, N., PERRIER, V., AND GOIRAND, E. 1996. Wavelets and turbulence. *Proceedings of the IEEE 84*, 4, 639–669.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. In *Proceedings of SIGGRAPH*.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH*, 15–22.

FRISCH, U. 1995. *Turbulence: The Legacy of A. N. Kolmogorov*. Cambridge University Press.

KIM, B., LIU, Y., LLAMAS, I., AND ROSSIGNAC, J. 2005. Flowfixer: Using BFECC for fluid simulation. In *Proceedings of Eurographics Workshop on Natural Phenomena*.

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. In *Proceedings of ACM SIGGRAPH*.

LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of flames for a production environment. In *Proceedings of ACM SIGGRAPH*.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *Proceedings of ACM SIGGRAPH*, 457–462.

MCNAMARA, A., TREUILLE, A., POPOVIC, Z., AND STAM, J. 2004. Fluid control using the adjoint method. In *Proceedings of SIGGRAPH*.

NEYRET, F. 2003. Advected textures. In *ACM SIGGRAPH/EG Symposium on Computer Animation (SCA)*.

PERLIN, K. 1985. An image synthesizer. In *Proceedings of ACM SIGGRAPH*, 287–296.

PERRIER, V., PHILIPOVITCH, T., AND BASDEVANT, C. 1995. Wavelet spectra compared to fourier spectra. *Journal of Mathematical Physics 36*.

RASMUSSEN, N., NGUYEN, D. Q., GEIGER, W., AND FEDKIW, R. 2003. Smoke simulation for large scale phenomena. In *Proceedings of ACM SIGGRAPH*.

SCOTTI, A., AND MENEVEAU, C. 1999. A fractal model for large eddy simulation of turbulent flows. *Physica D*, 198–232.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. In *Proceedings of SIGGRAPH*.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing (in press)*.

SHRAIMAN, B., AND SIGGIA, E. 2000. Scalar turbulence. *Nature*, 405, 639–646.

STAM, J., AND FIUME, E. 1993. Turbulent wind fields for gaseous phenomena. In *Proceedings of ACM SIGGRAPH*.

STAM, J. 1999. Stable fluids. In *Proceedings of ACM SIGGRAPH*.

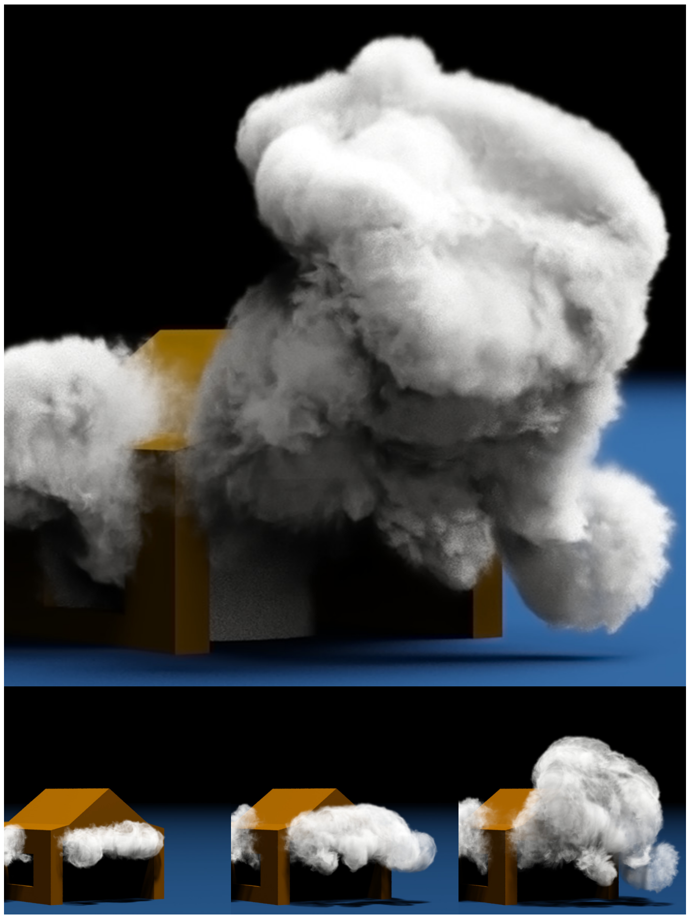

**Figure 3:** *Flow around a complex obstacle with grid-based densities. Wavelet turbulence synthesized a* $720 \times 576 \times 576$ *grid from a* $80 \times 64 \times 64$ *grid. Each frame took less than two minutes on an eight core workstation.*

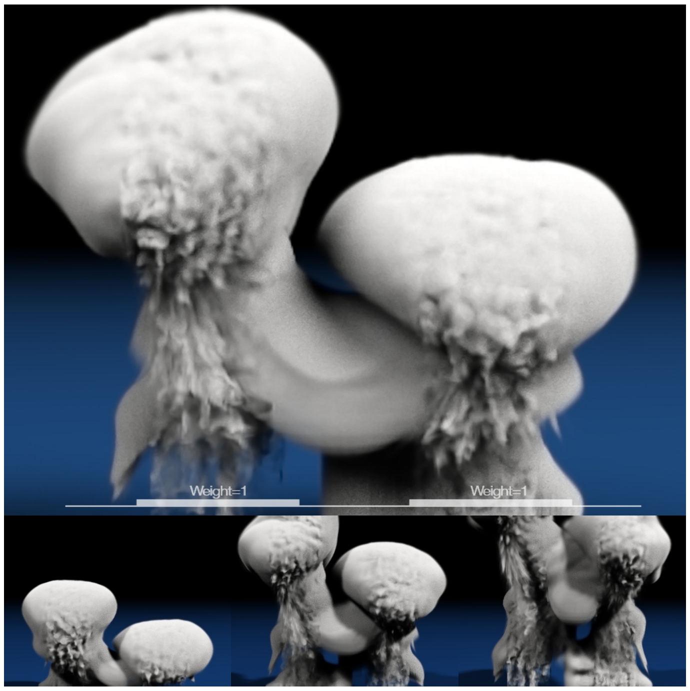

**Figure 4:** *The user can control turbulence arbitrarily. Here turbulence is activated in two vertical slabs, denoted on the bottom of the top image.*

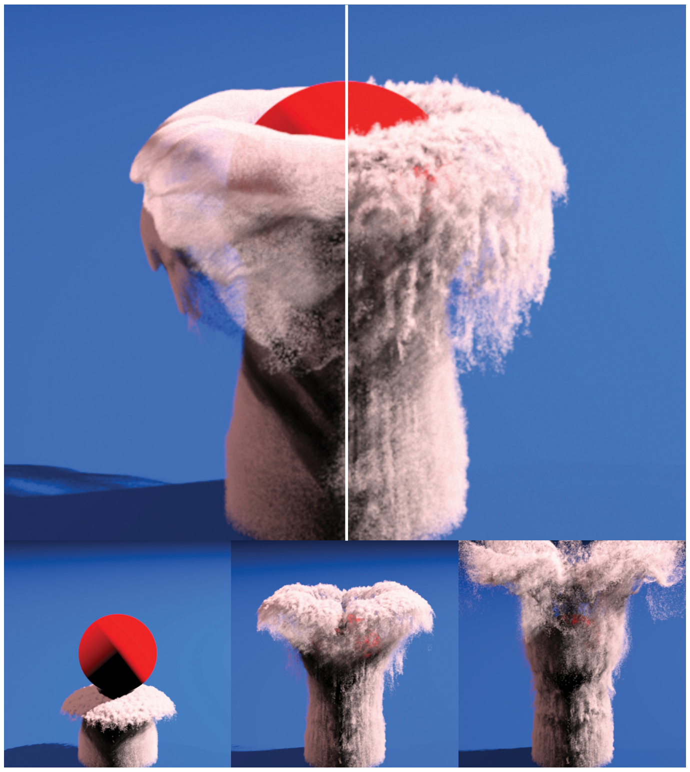

**Figure 5:** *Flow around a sphere with particle-based densities. The low resolution simulation is shown in the left half of the top image. Resolution and performance are identical to Fig. 1*

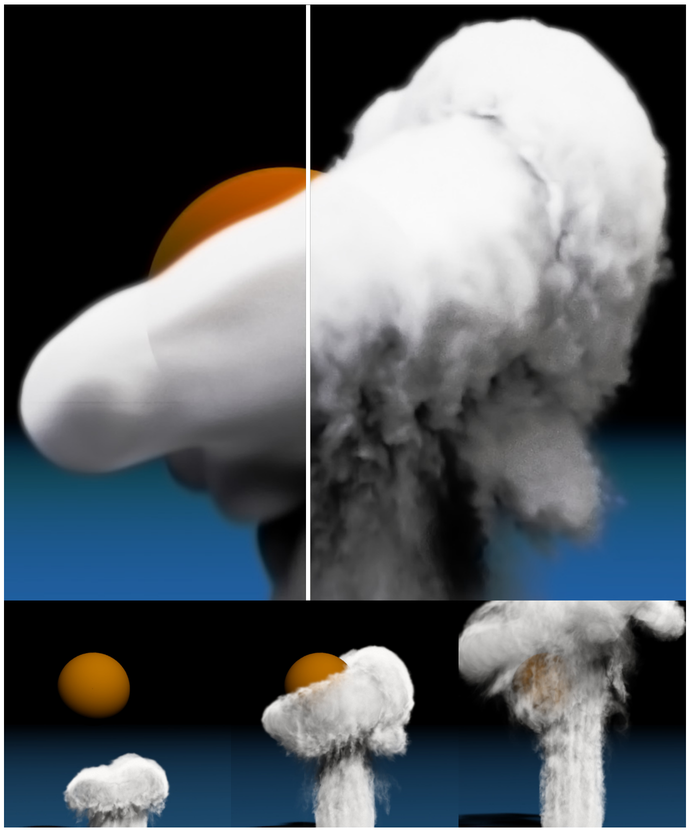

**Figure 6:** *Flow around a sphere with grid-based densities. The low resolution simulation is shown in the left half of the top image. Wavelet turbulence synthesized a* $400^3$ *grid from a* $50^3$ *grid. Each frame took an of average 30 seconds on a four core workstation.*