Wavelet Turbulence Improved

Paul Liu

ABSTRACT

The modeling of turbulence in fluids is essential in capturing the lively look of phenomenon such as smoke and fire. Unfortunately, highly turbulent flows require extremely fine grids to simulate accurately, and is prohibitive for graphics applications. With the Wavelet Turbulence method [4], procedurally generated turbulence can be added to coarse grid simulations. This is done by modeling turbulence above simulation scales as a time-constant incompressible field of isotropic noise. In this project, we attempt to refine the ideas of Wavelet Turbulence by supporting anisotropic noise as well as a time-varying noise field.

CCS CONCEPTS

• **Computing methodologies → Simulation by animation**; **Physical simulation**;

KEYWORDS

turbulence, wavelets, noise, fluids, simulation control, gabor

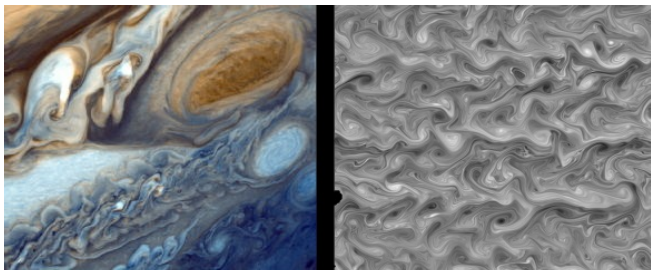


Figure 1: Examples of anisotropic turbulence. Left: Picture of jupiter taken from the voyager. Right: anisotropic beta-plane turbulence.

1 INTRODUCTION

To capture the lively look of phenomenon such as smoke and fire, the modeling of turbulence is essential. Accurate modeling of turbulent effects is possible through direct numerical simulation (DNS) of the Navier-Stokes equations over a grid. However, highly turbulent flows require extremely fine grids, and thus DNS is inadequate for graphics applications.

The Wavelet Turbulence (WT) method [4] allows a user to add turbulent details procedurally, using significantly less time than a full resolution simulation. In the WT method, a wavelet decomposition is used to determine where turbulent details are lost, and procedurally generated turbulent noise is added to compensate for lost details. This takes advantage of Kolmogorov's observation that small-scale turbulence have statistics similar to random noise [10], and can thus be mimicked by a an appropriate noise function.

The noise function used in WT combines the idea of Curl Noise [1] and Wavelet Noise [2] to create a band-limited, isotropic, and incompressible noise field to model turbulence. Due to the flexibility of user-control in WT, the method has been extremely successful. However, there two main drawbacks to WT: (1) the assumption of isotropic noise is not valid in all situations, such as when the fluid is flowing past obstacles. (2) The turbulence noise field used in WT is not time varying, and subtle visual artifacts can occur as a result.

Contributions. We propose two improvements to wavelet turbulence that allows for the modeling of time-varying anistropic turbulence. We model the stress tensor of the turbulence in each grid cell with anisotropic turbulence models from the computational fluid dynamics community [10]. Given the stress tensor, we have a preferred direction and rotation for the turbulence, for which we can model with anisotropic gabor noise. To create a time-varying effect, we modify the noise generation method of Lagae et al. [5] to include a time parameter. Our modified procedure produces a sequence of temporally coherent noise tiles that all have the frequency statistics. The algorithms we propose are easy to implement, and introduce only a small overhead to the memory and simulation time.

2 RELATED WORK

Vorticity and turbulence effects were sought after since the early days of fluid simulation in graphics. To preserve turbulence in the presence of viscosity, techniques such as vorticity confinement [3] and vortex particles [8] were developed. However, these methods are only capable of preserving turbulence that are of equal or greater scale to the simulation grid.

Often practitioners want to simulate a fluid on a coarse grid, and then add extra subgrid details after. In the domain of subgrid turbulence, methods such as Wavelet Turbulence (WT) [4], the 4D Kolmogorov method (4DKM) [9], and Curl Noise (CN) [1] are available. Of these methods, WT is perhaps currently the most commonly used. At the heart of these subgrid methods, a scalar noise function is required. WT uses Wavelet Noise, 4DKM uses filtered white noise, and CN uses Perlin Noise. A thorough comparison of these noise functions can be found in Lagae et al. [5]. Recently, Gabor Noise [5] was developed for both anisotropic and isotropic noise field generation.

For the most part, researchers have focused on generating isotropic turbulence, often with little attention paid to anisotropic effects. These effects are subtlely present in most fluid simulations, especially near obstacle boundaries (see Figure 1). In the area of anisotropic turbulence modeling, Pfaff et al. [6, 7] were able to exhibit various anisotropic effects using the popular $k - \epsilon$ from the computational fluid dynamics community.

3 BACKGROUND

3.1 Wavelet Turbulence

In Wavelet Turbulence (WT), the user first begins with a coarse fluid simulation. The coarse scale velocities are then interpolated and added onto a high-resolution grid of turbulent noise. The turbulent noise is static in time, and generated in a way consistent with

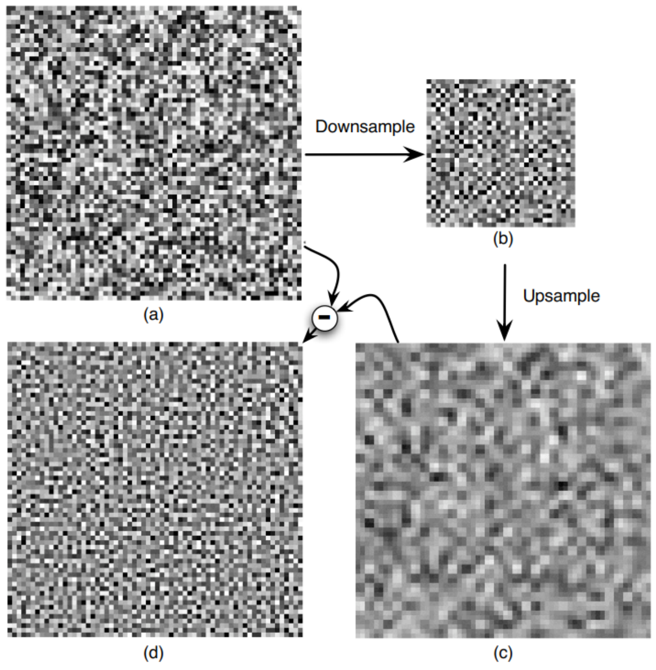


Figure 2: Generating a wavelet noise tile.

Komolgorov's theory of turbulence [4]. In this way, the simulation resolution is amplified, and missing details are procedurally added by the noise field. To determine the strength of the turbulent noise at each grid cell, a wavelet decomposition is done on the coarse grid simulation to compute where turbulent energy is lost. A key component of WT is the generation of the turbulent noise grid. To create the turbulent noise field, WT first takes three time-constant scalar wavelet noise fields w_1, w_2, w_3, and creates a 3D noise field Y by taking

$$??W = \left(\frac{\partial w_1}{\partial y} - \frac{\partial w_2}{\partial z}, \frac{\partial w_3}{\partial z} - \frac{\partial w_1}{\partial x}, \frac{\partial w_2}{\partial x} - \frac{\partial w_3}{\partial y}\right). \qquad (1)$$

Note that Y is incompressible since it is a sum of curl fields. Thus it can be added to existing fluid simulations without violating incompressibility constraints. Given Y, we can generate a turbulent noise field W consistent with Komolgorov theory by taking sums of the noise field at various frequencies:

$$W(x) = \sum_{f=f_{min}}^{f=f_{max}} Y(2^i x) 2^{-\frac{5}{6}(f-f_{min})}. \qquad (2)$$

More details regarding the derivation can be found in [4].

In WT, wavelet noise is used for the w_i's. The key property of wavelet noise is that it is approximately band-limited, ensuring that procedurally generated turbulence will not interfere with frequencies that are resolved by the coarse grid simulation.

To generate the wavelet noise fields, Kim et al. [4] use the procedure of Cook and DeRose [2] (shown in Figure 2). The noise field is created by first generating uniform noise on a grid. This noise is then down-sampled and up-sampled to remove frequencies above

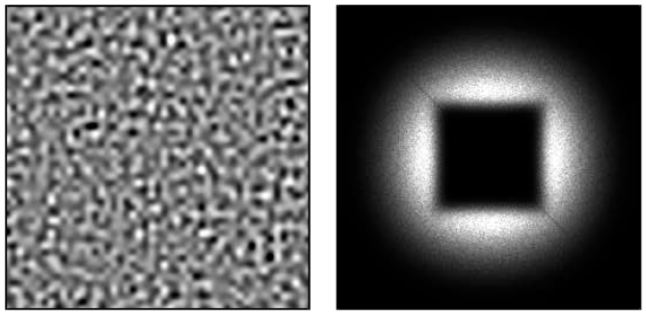


Figure 3: Wavelet noise in the spatial (left) and frequency (right) domain.

the grid resolution. Finally, the resampled noise tile is subtracted from the original noise tile to create a band-limited isotropic noise.

A figure of wavelet noise in the frequency domain can be found in Figure 3. Note that although wavelet noise is band-limited, it suffers from axis-aligned artifacts due to the fact that it is generated on a grid.

3.2 Gabor Noise

In practice, any band-limited scalar noise can be fed into Equation **??** to generate an incompressible noise field. This motivates the search for a more appropriate noise field for turbulence, especially one that supports anisotropic effects and without axis-aligned artifacts.

Both requirements can be addressed by using the gabor noise of Lagae et al. [5]. In gabor noise, one constructs the noise field with a sum of gabor kernels,

$$g(x, y) = K \exp\left(-\pi a^2(x^2 + y^2)\right) \cos\left(2\pi F_0 \left(x \cos \omega_0 + y \sin \omega_0\right)\right).$$

The gabor kernel g is gaussian modulated in both the spatial and frequency domain, with the position in the frequency domain controlled by parameters F_0 and ω_0. Due to this, one can create band-limited noise by randomly placing gabor kernels within the frequency band of noise that we want to generate (see [5] for details). One such example is given in Figure 4, where gabor kernels were randomly placed within an annulus in the spectral domain. Note that since we have analytic expressions for g and its fourier transform, we do not need to perform any costly spectral operations to place the gabor kernel in frequency space. We also note that the gabor kernel can easily be generalized to higher dimensions if necessary.

By place the gabor kernels in the appropriate configuration, one can create anisotropic noise patterns. Examples of generated anisotropic noise can be found in Figure 5.

4 IMPROVEMENTS TO WAVELET TURBULENCE

4.1 Dynamic Gabor Noise

To create a dynamic time-varying noise, one can start with an initial configuration of gabor kernels, and perform rigid rotations of the kernels in frequency space to get a gabor noise field that varies in time. In all of our examples, the configuration of the gabor

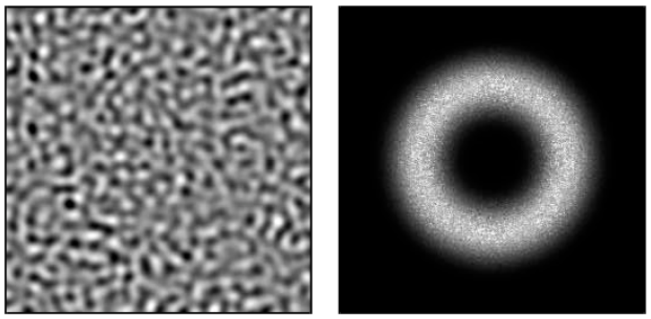


Figure 4: Isotropic gabor noise in the spatial (left) and frequency (right) domain.

kernels have local radial symmetry in some way. If we perform the rotations in a way that respects the symmetry of the noise kernels, every noise field generated from the rotations will follow approximately the same frequency spectrum. We elaborate on this in several examples below.

Isotropic band-limited noise. For isotropic band-limited noise, the gabor kernels are always placed in an annulus (see Figure 4). Thus to get coherent time-varying noise, we can simply rotate the annulus about its centre (see Figure 6).

Anisotropic band-limited noise. For anisotropic band-limited noise, the gabor kernels are always placed in two symmetric lobes (see Figure 4). Thus to get coherent time-varying noise, we can simply rotate the lobes about their respective centres (see Figure 6).

Videos of dynamic isotropic and anisotropic noise are attached.

4.2 Anisotropic Noise Models

As wavelet turbulence is generated from an isotropic noise field, anisotropic subgrid turbulence cannot be produced. This lack of anistropy can be most clearly seen near obstacle boundaries where shear effects are prominent. The effect manifests as excessive turbulence spreading in all directions, instead of turbulence confined to a layer (see Figure 8).

To model the anisotropy, we observe that the shear effects of the flow is captured within the local strain rate tensor. Let $u(x_1, x_2, x_3) = (u_1, u_2, u_3)$ be the velocity field of the fluid at some time. Then the local strain rate tensor is a 3×3 matrix S, where

$$S_{ij}(x_1, x_2, x_3) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

At each point (x_1, x_2, x_3), we locally transform the turbulence noise field W with the normalized local strain rate tensor:

$$\tilde{W} = \frac{k}{||S||_F} SW,$$

where k is the highest frequency energy resolved from the coarse simulation at point (x_1, x_2, x_3), as is done in wavelet turbulence.

5 IMPLEMENTATION AND RESULTS

In our implementation we experimented with two different fluid simulation frameworks. The dynamic gabor noise was implemented within the fluid simulation code of Kim et al. [4], and the anisotropic

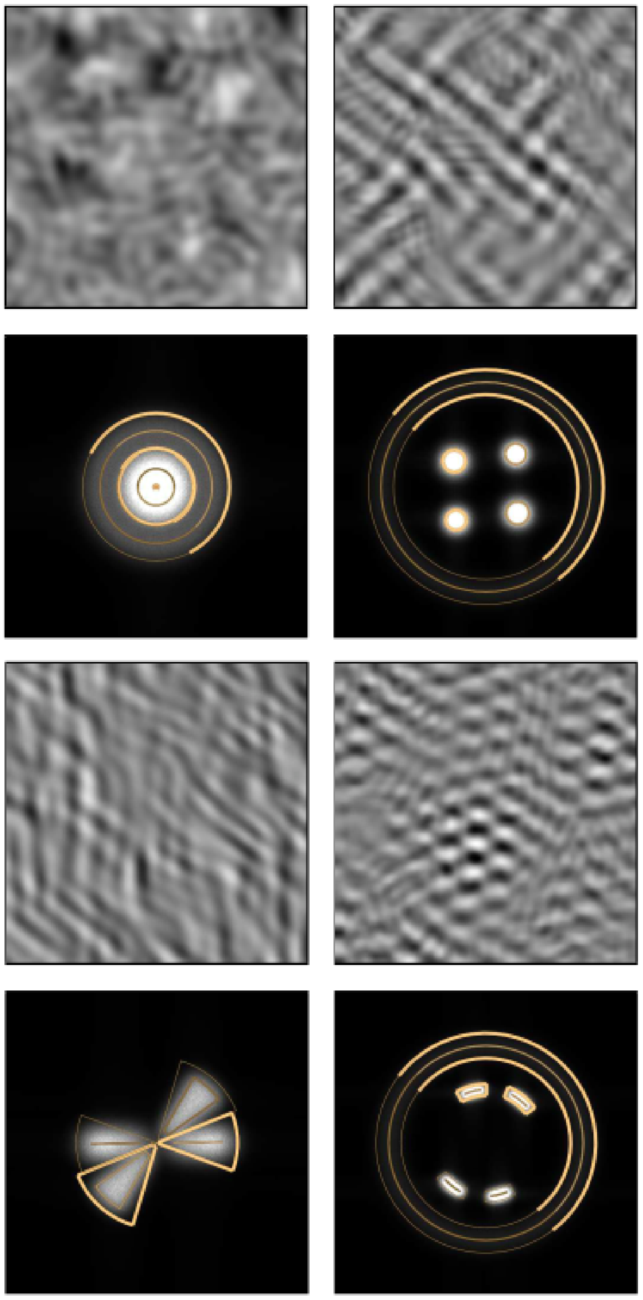


Figure 5: Examples of anisotropic gabor noise in spatial and frequency domains.

turbulence model was implemented within Manta. These choices were made for implementation convenience. Manta had a pre-existing framework for implementing stress models, while the wavelet turbulence code of Kim et al. enabled easy comparisons between wavelet noise and gabor noise.

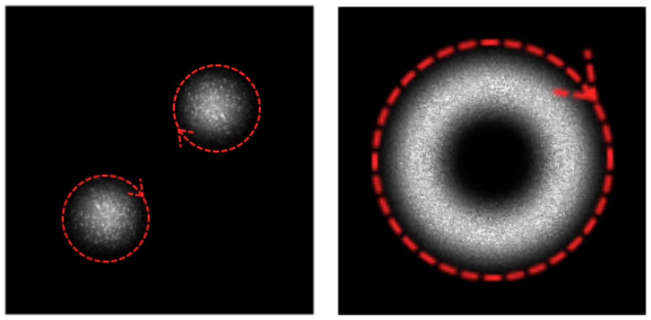


Figure 6: Creating time-varying isotropic (left) and anisotropic (right) gabor noise. The gabour noise kernels are rotated rigidly in the regions contained within the red circles. Due to radial symmetry of the configuration, the spectrum of the gabor noise stays approximately the same, but a time-varying effect is created.

Dynamic Gabor Noise. To implement our dynamic gabor noise, we pre-generated several gabor noise tiles as described in Section 4.1. Although these noise tiles can be computed on the fly (since each noise tile is encoded by a single seed value and a rotation angle), we found it faster and more convenient to pre-evaluate and cache the tiles. The memory cost of the procedural turbulence algorithm increases slightly, but is negligible compared to the cost of storing the simulation grids. On each frame we linearly interpolate from noise tile to noise tile and get a continuous time-varying noise (as shown in the accompanying videos). Figure 7 shows a comparison of static wavelet noise versus dynamic gabor noise. In this example, fluid is continuously injected into the centre of the simulation grid. The static wavelet noise field tends to collect smoke in fixed regions, leading to a stagnant appearance in the simulation. The dynamic noise on the other hand, remains lively throughout. Videos of both simulations are attached.

Anisotropic Noise Model. Our anistropic noise model was implemented as a Manta plugin. The evaluation cost of our model is negligible as the pressure and velocity solves dominate the running time. Figure 8 shows a particle based simulation of (1) wavelet turbulence, (2) the $k - \epsilon$ of [6], and (3) our anisotropic stress model. Both the $k - \epsilon$ model [6] and our anisotropic model capture the shear effects near the obstacle, whereas normal wavelet turbulence diffuses the particles too much near the obstacle boundary. Unfortunately, the rotation by the strain tensor also seems to make the noise field too incompressible, as artifacts can clearly be seen in the simulation. Code to run the simulations and videos are available on request (we opted not to upload the videos as it made the project submission too large and this project was never going to upload before midnight with it).

6 FUTURE WORK

Although we were able to capture anisotropic shear effects, we were not able to produce a model that captures anisotropic effects in the frequency domain. If we could compute anisotropy in the frequency domain, we can generate matching anisotropic gabor noise. One example of such anisotropy is the Kármán vortex sheet shown in Figure 9. In this example, flow past the cylindrical obstacle generates alternating vortices above and below the cylinder. Interestingly, the vortices above the cylinder always rotate clockwise, and the vortices below the cylinder always rotates counter-clockwise. If we applied either wavelet turbulence or our anisotropic stress model to this example, the noise tiles would create vortices swirling in the wrong direction above and below the cylinder. In the future, we hope to develop an anisotropic turbulence that model that accounts for this and other types of frequency anisotropy.

REFERENCES

[1] Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-noise for procedural fluid flow. *ACM Trans. Graph.* 26, 3 (2007), 46. https://doi.org/10.1145/1276377.1276435
[2] Robert L. Cook and Tony DeRose. 2005. Wavelet noise. *ACM Trans. Graph.* 24, 3 (2005), 803–811. https://doi.org/10.1145/1073204.1073264
[3] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001.* 15–22. https://doi.org/10.1145/383259.383260
[4] Theodore Kim, Nils Thürey, Doug L. James, and Markus H. Gross. 2008. Wavelet turbulence for fluid simulation. *ACM Trans. Graph.* 27, 3 (2008), 50:1–50:6. https://doi.org/10.1145/1360612.1360649
[5] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural noise using sparse Gabor convolution. *ACM Trans. Graph.* 28, 3 (2009), 54:1–54:10. https://doi.org/10.1145/1531326.1531360
[6] Tobias Pfaff, Nils Thürey, Jonathan M. Cohen, Sarah Tariq, and Markus H. Gross. 2010. Scalable fluid simulation using anisotropic turbulence particles. *ACM Trans. Graph.* 29, 6 (2010), 174:1–174:8. https://doi.org/10.1145/1882261.1866196
[7] Tobias Pfaff, Nils Thürey, Andrew Selle, and Markus H. Gross. 2009. Synthetic turbulence using artificial boundary layers. *ACM Trans. Graph.* 28, 5 (2009), 121:1–121:10. https://doi.org/10.1145/1618452.1618467
[8] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.* 24, 3 (2005), 910–914. https://doi.org/10.1145/1073204.1073282
[9] Jos Stam and Eugene Fiume. 1993. Turbulent wind fields for gaseous phenomena. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993, Anaheim, CA, USA, August 2-6, 1993.* 369–376. https://doi.org/10.1145/166117.166163
[10] Henk Kaarle Versteeg and Weeratunge Malalasekera. 1995. *An introduction to computational fluid dynamics - the finite volume method.* Addison-Wesley-Longman.

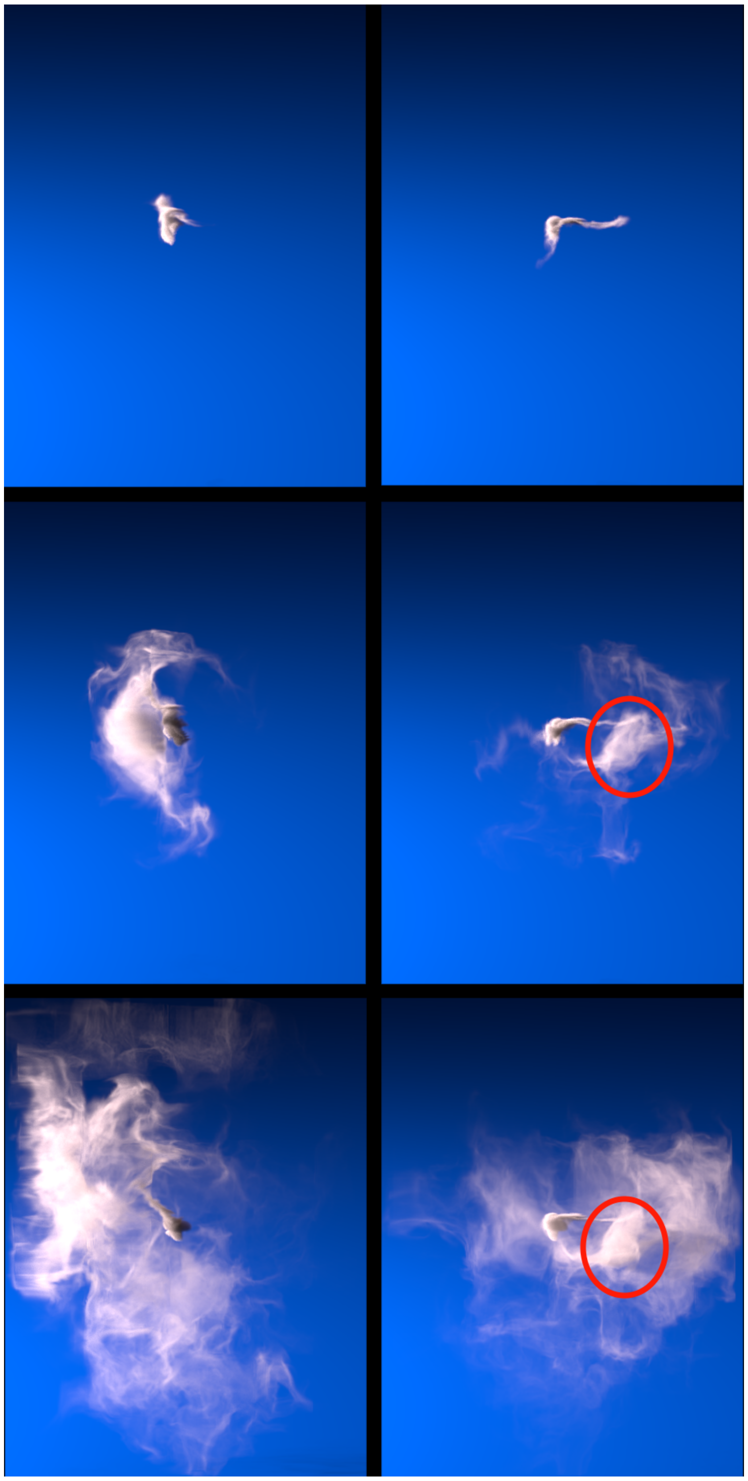
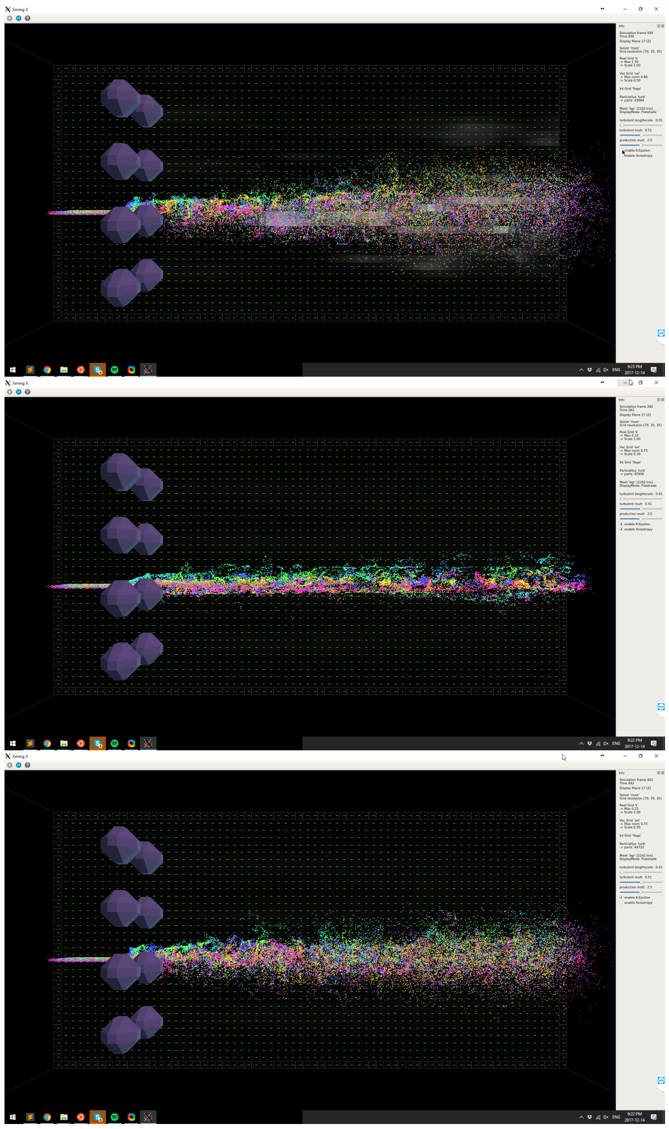


Figure 7: Smoke simulation for a time-varying (left column) gabor noise and a time-constant (right column) wavelet noise after 10 (top), 100 (middle), and 300 (right) time steps. Note that the region outlined in red stays stagnant throughout the 300 frames for wavelet noise, whereas gabor noise diffuses the smoke more actively.



Simulation frame 930
Time 930
Display Plane 17 [Z]
Solver 'main'
Grid resolution [70, 35, 35]
Real Grid 'k'
-> Max 1.50
-> Scale 1.00
Vec Grid 'vel'
-> Max norm 0.80
-> Scale 0.50
Int Grid 'flags'
ParticleSys 'turb'
-> parts: 43664
Mesh 'bgr' [2242 tris]
DisplayMode: Flatshade
turbulent lengthscale: 0.01
turbulent mult: 0.51
production mult: 2.5
☐ enable K-Epsilon
☐ enable Anisotropy

Simulation frame 282
Time 282
Display Plane 17 [Z]
Solver 'main'
Grid resolution [70, 35, 35]
Real Grid 'k'
-> Max 0.15
-> Scale 1.00
Vec Grid 'vel'
-> Max norm 0.75
-> Scale 0.50
Int Grid 'flags'
ParticleSys 'turb'
-> parts: 45906
Mesh 'bgr' [2242 tris]
DisplayMode: Flatshade
turbulent lengthscale: 0.01
turbulent mult: 0.51
production mult: 2.5
✓ enable K-Epsilon
✓ enable Anisotropy

Simulation frame 442
Time 442
Display Plane 17 [Z]
Solver 'main'
Grid resolution [70, 35, 35]
Real Grid 'k'
-> Max 0.15
-> Scale 1.00
Vec Grid 'vel'
-> Max norm 0.75
-> Scale 0.50
Int Grid 'flags'
ParticleSys 'turb'
-> parts: 44722
Mesh 'bgr' [2242 tris]
DisplayMode: Flatshade
turbulent lengthscale: 0.01
turbulent mult: 0.51
production mult: 2.5
✓ enable K-Epsilon
☐ enable Anisotropy

Figure 8: Procedural turbulence for an anisotropic example. Wavelet turbulence (top), our anisotropic stress model (middle), $k - \epsilon$ model (bottom). The isotropic wavelet turbulence clearly spreads out the fluid particles too much, in contrast to the $k - \epsilon$ model and the anisotropic stress model.

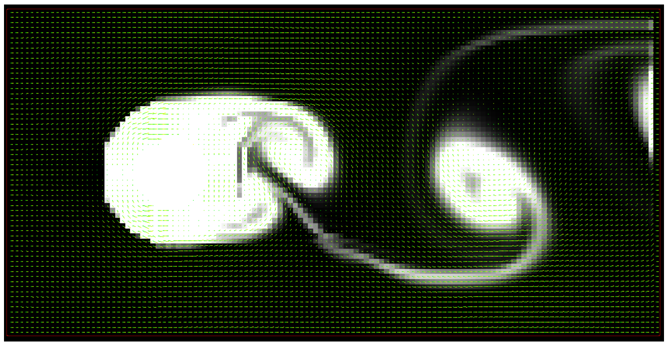


Figure 9: Kármán vortex sheet.