Scalable Fluid Simulation using Anisotropic Turbulence Particles

Tobias Pfaff[1] Nils Thuerey[1] Jonathan Cohen[2] Sarah Tariq[2] Markus Gross[1]

[1] ETH Zurich [2] NVIDIA

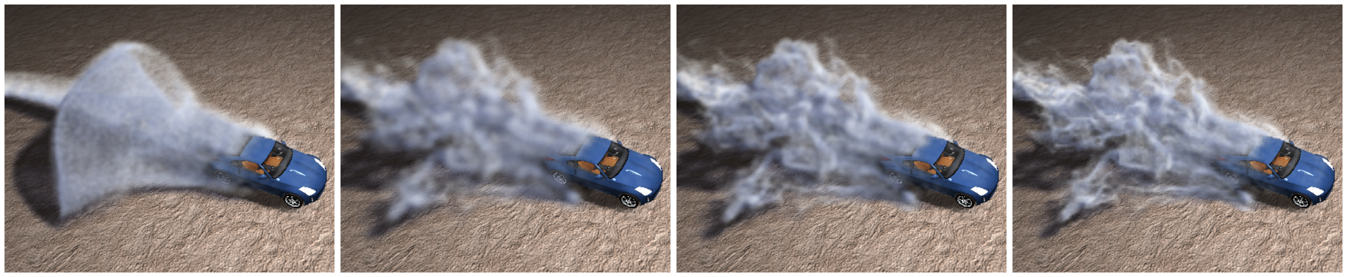

Figure 1: Our method easily scales from an inexpensive, coarse fluid simulation to a detailed turbulent wake. The base solver, shown to the left uses only $32 \times 8 \times 32$ cells. The remaining pictures show the influence of our turbulence model, with a varying number of particles from 250k to 1M and 4M from left to right. For the simulation with 1M particles we achieve 15 frames per second on average, including rendering. While the amount of detail directly depends on the number of particles used, the overall flow remains consistent.

Abstract

It is usually difficult to resolve the fine details of turbulent flows, especially when targeting real-time applications. We present a novel, scalable turbulence method that uses a realistic energy model and an efficient particle representation that allows for the accurate and robust simulation of small-scale detail. We compute transport of turbulent energy using a complete two-equation k–ε model with accurate production terms that allows us to capture anisotropic turbulence effects, which integrate smoothly into the base flow. We only require a very low grid resolution to resolve the underlying base flow. As we offload complexity from the fluid solver to the particle system, we can control the detail of the simulation easily by adjusting the number of particles, without changing the large scale behavior. In addition, no computations are wasted on areas that are not visible. We demonstrate that due to the design of our algorithm it is highly suitable for massively parallel architectures, and is able to generate detailed turbulent simulations with millions of particles at high framerates.

Keywords: Turbulence, Physically Based Animation, Fluid Simulation

1 Introduction

The phenomena of smoke, water and fire are fascinating to watch, one of their most remarkable characteristics being their chaotic, turbulent nature. In order to create a convincing animated movie or interactive application with fluid effects, it is therefore vital to capture this turbulent behavior in a simulation. As turbulence in fluids extends over many scales, a direct simulation resolving all details requires a costly high resolution calculation, which is infeasible for real-time applications. The high cost of a direct numerical simulation has lead to increasing interest in algorithms to synthetically generate turbulence for augmenting low resolution simulations. When aiming at short production cycles or even interactivity, it is moreover essential that the method scales well across different application scenarios, from computer games to off-line animations. In addition, it is highly desirable that the large scale behavior is consistent when changing the amount of detail in the simulation.

Unfortunately, current turbulence methods, especially complex ones that capture effects more accurately, often rely on a strong coupling to the base simulation, which makes it hard to obtain predictable results for varying levels of detail. Additionally, current methods often have to make many assumptions about the production of turbulence. The production term is, however, a crucial factor that strongly determines the quality of the dynamics generated with the turbulence model later on. We will make use of a full two-equation energy transport model with physically plausible production terms, which also allow us to capture anisotropic effects. Anisotropy is important to ensure that the synthetically generated turbulence integrates into the base flow without unnatural disturbances.

We have designed our method to yield turbulent detail using a Lagrangian representation without requiring any neighborhood information. The latter is important to allow for an efficient calculation of detailed fluid motion on modern computing architectures such as multi-core CPUs, or massively parallel architectures such as GPUs, where inter-particle communication requires synchronization that can adversely impact scaling to wider architectures. The amount of detail to be resolved can be controlled by adjusting the number of particles with respect to the computation time available for each frame, while retaining a consistent large-scale flow.

Our approach is based on a separation of the large scale dynamics from the small scale turbulence: large scales are computed using a low resolution fluid solver, while the turbulent detail with anisotropic effects is computed on the particle system. As our energy transport model is only loosely coupled to the large scale flow, it allows us to use very coarse base simulations that are cheap to compute. Our contributions are as follows:

- a *scalable, particle based turbulence* model that is designed to work without particle-particle interaction, and is therefore

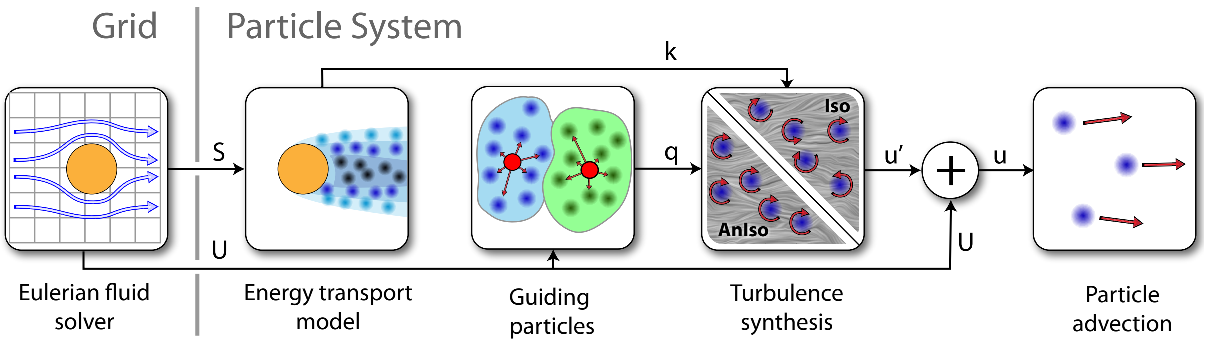

Figure 2: An overview of our algorithm. A low resolution grid-based solver provides a base velocity and strain field. For each particle, a turbulence model is computed, which drives the turbulence synthesis with isotropic and anisotropic turbulence. The particles velocity is given by the large scale velocity from the grid and the small scale turbulent velocity.

suitable for massively parallel computations;

- a robust *energy transport model* that realistically captures the production of turbulence,
- and a method for *anisotropic synthesis* that efficiently captures directional vortices, thereby realistically integrating the turbulence into the base flow.

2 Related Work

Fluid simulations have become popular in Computer Graphics with the introduction of the semi-Lagrangian stable fluids solver by Stam [1999]. Extensions of this basic solver have been made, e.g., to simulate liquids [Enright et al. 2002], bubble flows [Hong and Kim 2003], viscoelastic fluids [Goktekin et al. 2004], or interactions with rigid bodies [Carlson et al. 2004], to give only a few examples from the large body of work. In this work we will focus on single phase fluid simulations with Eulerian fluid solvers due to their widespread use.

Since the introduction of the stable fluids approach, researchers have fought with the problem of representing small scale detail without a costly high resolution simulation. One part of the problem is the inherent damping of turbulence detail due to numerical dissipation. A popular approach to alleviate this problem is the use of higher order advection schemes such as Back and Forth Error Correction [Kim et al. 2005], MacCormack advection [Selle et al. 2008], QUICK [Molemaker et al. 2008], FLIP [Zhu and Bridson 2005] or the family of CIP methods [Kim et al. 2008a]. Mullen et al. [2009] even propose an integration scheme that is guaranteed to preserve energy. While these methods significantly improve the properties of a basic simulation, the turbulent detail that they can represent is still inherently limited by the resolution of the underlying grid. Adaptive grid methods [Losasso et al. 2004], [Feldman et al. 2005], [Irving et al. 2006] address the problem by refinement. The computational overhead introduced by the adaptivity typically only pays off when the detailed motion is confined to only a small part of the simulation space, or for very high resolutions.

Another class of methods use synthetic turbulence instead of direct simulation of the small scale features, which allows detailed effects beyond the grid resolution. The first methods synthesized a divergence-free turbulence field using the Kolmogorov spectrum, e.g, in [Stam and Fiume 1993], [Lamorlette and Foster 2002], and [Rasmussen et al. 2003]. Selle et al. [2005] proposed the use of manually seeded vortex particles as a turbulence representation. [Bridson et al. 2007], on the other hand, suggested taking the curl of vector noise fields to produce divergence free velocity fields. [Zhao et al. 2010] propose the use of random forcing to integrate a statistical turbulence field with a given simulation.

However, these methods do not take into account the spatial distribution and dynamics of the underlying turbulence. Therefore, recent methods use a more complex estimation of flow statistics to better capture the characteristics of turbulence. Kim et al. [2008b] use wavelet decomposition to determine local turbulence intensities. This approach assumes that the base solver can resolve the turbulence dynamics, which is not always the case, e.g., for complex scenes or very coarse solver resolutions. In our model, we will make use of wavelet turbulence for synthesis, but improve the coupling with the fluid simulation and energy transport between different scales of turbulence. Similarly, Schechter [2008], Narain [2008], and Pfaff et al.[2009] use energy transport models to derive turbulence parameters. While this improves the turbulence dynamics, they need to significantly simplify the energy transport, as in Schechter [2008], or make strong assumptions, such as mixing-length models to close their one-equation energy model [Narain et al. 2008]. The method we propose below is based on a stabilized version of the full two-equation model, and thus physically more accurate, especially with respect to the production terms. While the former two primarily target isotropic turbulence, Pfaff et al.[2009] were able to represent anisotropic effects near obstacles. However, the method cannot handle free stream turbulence such as rising smoke, and requires careful tuning for the particle decay in order to obtain fine-grained turbulence. Moreover, it is not suitable for real-time application as it relies on inter-particle interactions.

There are only few methods that enable detailed fluid simulation at interactive frame rates. Crane [2007] demonstrated the realization of three-dimensional Eulerian fluid solvers on a GPU, while Cohen et al. [2010] use a multigrid GPU based-solver to efficiently solve the Navier-Stokes equations in real-time. However, fine-grained turbulent detail is hard to achieve with these direct approaches. We use the approach of Cohen [2010] for an underlying Eulerian solver, and use its information to drive our particle based turbulence model. Wicke et al. [2009] presented a method to precompute and couple reduced bases of flows, enabling large scale simulations at real-time frame rates. As the method requires large amounts of memory for complex scenes, it is difficult to apply in interactive scenarios. Horvath et al. [2009] use a hybrid particle approach with a coupled 2D and 3D simulation to efficiently simulate fire simulations on the GPU. However, their approach targets scenes with a fixed camera perspective. Real-time simulations of particle based liquids have been demonstrated in [Müller et al. 2005], but as these simulations heavily depend on neighborhood calculations, detailed simulations can be very expensive, and can be difficult to stabilize. To summarize, few existing approaches are suitable for simulating small scale turbulence at interactive frame-rates, as they either rely on grids with high resolutions, or do not scale well due to a strong coupling to the base simulation.

3 Turbulence Model

To efficiently simulate small scale turbulence it is not feasible to directly represent the turbulent motion using a high resolution velocity grid, as the computational effort increases strongly with grid resolution. Instead, we describe the turbulence field by its statistical properties, and synthesize turbulence only where needed. Our simulation approach is driven by a low-resolution Eulerian fluid solver, and a particle system. The particles coincide with the smoke particles used for rendering, while the grid-based solver is used to obtain the large scale characteristics of the flow. The turbulence is computed and synthesized directly on the particles, each of which is influenced by a texture based turbulence representation, and stores a preferred axis of rotation for anisotropic effects. An overview of our model is given in Fig. 2. We chose to describe the turbulence using an energy representation, as this allows us to adapt powerful transport models for our simulations. The spatial and temporal energy distributions driving the turbulence synthesis are obtained using a modified k–ε turbulence model that we will explain in the following sections.

3.1 Energy transport

To simulate the energy dynamics, we use the k–ε model by Launder and Sharma [1974]), which is one of the most widely used turbulence models in CFD. It is a complete two-equation model, which unlike one-equation models requires no additional problem-defendant assumptions, such as the mixing length. On the basis of a large scale flow field $\mathbf{U}$, it models the two variables k and ε on an averaged large scale. While k represents the turbulent kinetic energy contained in the smaller scales, ε stands for the dissipation of the turbulence structures. A thorough discussion of different turbulence models used in CFD can be found in the books of Wilcox [1993] and Pope [2000]. The k–ε model consists of two coupled partial differential equations that specify the evolution of the turbulent kinetic energy k

$$\frac{\partial k}{\partial t} + \mathbf{U}\nabla k = \nabla\left(\frac{\nu_T}{\sigma_1}\nabla k\right) + P - \varepsilon \tag{1}$$

and the turbulence dissipation ε :

$$\frac{\partial \varepsilon}{\partial t} + \mathbf{U}\nabla \varepsilon = \nabla\left(\frac{\nu_T}{\sigma_2}\nabla \varepsilon\right) + \frac{\varepsilon}{k}\left(C_1 P - C_2\varepsilon\right) . \tag{2}$$

Eq. (1) and Eq. (2) share the same structure: the left-hand side contains an advection in the mean flow field. The right-hand side of both equations consist of a viscous diffusion term, a production and a dissipation term, in that order. The turbulent viscosity ν_T is a virtual viscosity, describing the effect of the small-scale turbulent motion as a viscous diffusive effect on the averaged, large scale of the model. Turbulent viscosity is defined as

$$\nu_T = C_\mu \frac{k^2}{\varepsilon} . \tag{3}$$

$C_{1,2}$, $\sigma_{1,2}$ are modeling constants with the standard values $C_\mu = 0.09$, $C_1 = 1.44$, $C_2 = 1.92$, $\sigma_k = 1$ and $\sigma_\varepsilon = 1.3$ according to [Launder and Sharma 1974].

The production P, i.e. energy transfer from the large scale flow field $\mathbf{U}$ to small scale turbulence, is defined in terms of the strain of the coarse flow S_{ij} as

$$P = 2\nu_T \sum_{ij} {S_{ij}}^2 , \tag{4}$$

with $S_{ij} = \frac{1}{2}(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i})$. Given values for k and ε this equation relates the strain of the fluid to the production of turbulence without the need for further assumptions about the flow. In addition, this formulation allows us to easily determine the amount of anisotropic turbulence that is produced, as will be described in § 3.3.

Instead of discretizing and solving these PDEs on a Eulerian grid, we compute them directly on the particle system. In this Lagrangian setting, Eq. (1) and Eq. (2) simplify. The advection is inherently handled by the motion of the particles with the flow. The effects of the diffusive terms in the $k-\varepsilon$ equations is to mix turbulent energy spatially. Since the turbulent particle motion causes mixing as particles cross paths, we find incorporating additional diffusion to be visually unnecessary. Avoiding these terms also allows us to track k and ε independently for all particles and skip a costly communication step. This yields the following equations for our model:

$$\frac{\mathrm{D}k}{\mathrm{D}t} = P - \varepsilon \tag{5}$$

$$\frac{\mathrm{D}\varepsilon}{\mathrm{D}t} = \frac{\varepsilon}{k}(C_1 P - C_2\varepsilon). \tag{6}$$

3.2 Turbulence Synthesis

Turbulent energy is usually studied with respect to the spatial scales of the structures in the flow field. The production of turbulent energy is typically concentrated in the energy-containing range of a fluid, its large scales, while dissipation to heat is growing stronger for the small scales. In between these two extrema lies the so called inertial subrange, in which the predominant energy transport mechanism is forward-scattering, transporting the energy from large to small scales. This transfer process can be modeled with time dependence, e.g. using the transient model of Obukhov [1941]. However, the model is often not practical, as the exponential nature of the transfer terms requires a high spectral and temporal resolution for a stable solution [Panchev 1971]. Fortunately the transfer phenomena in a flow quickly drive the distribution to a stationary solution in all but extreme situations. Therefore, practical approaches typically focus on the stationary solution only. Our method concentrates on scales mostly within the inertial subrange for which the well known Kolmogorov five-thirds law, as in [Frisch 1995], is a reasonable approximation. We will first describe our approach to compute isotropic turbulence, and then explain our extension for anisotropic effects.

To synthesize isotropic turbulence, the spectrum is divided into N octaves. In all our demos, we uses three octaves. Similar to [Kim et al. 2008b], each band is synthesized using the curl of band-limited wavelet noise [Cook and DeRose 2005] with band coefficients determined using a five-thirds law. The total velocity $\mathbf{u}$ of a particle is then given by the large scale flow velocity $\mathbf{U}$, interpolated from the low resolution Eulerian solver, and the turbulence velocity:

$$\mathbf{u} = \mathbf{U} + 2(\alpha k)^{\frac{1}{2}} \sum_i^N \omega_\mathbf{i}(\mathbf{q})\, 2^{-\frac{5}{6}i} \tag{7}$$

Here, ω is the curl noise textures, evaluated as described in Kim et al. [2008b], and $\mathbf{q}$ represents a texture coordinate stored for each particle, and k denotes the energy of the largest-scale turbulence band. The scaling of the wavelet turbulence is chosen such that the largest synthesized vortices cover 2–4 grid cells, as vortices on these scales are usually dampened out by numerical viscosity of the Eulerian simulation which is better able to represent larger vortices. The energy for the synthetic turbulence is directly given by the $k-\varepsilon$ model with $k = k_{iso} + k_{an}$ for each particle, where k_{iso} denotes the isotropic energy, and k_{an} the anisotropic energy. Assuming $k_{an} = 0$ for now, the energy for the largest band is given by $\alpha\, k_{iso}$. The scaling parameter α encodes the shape of the assumed energy spectrum,

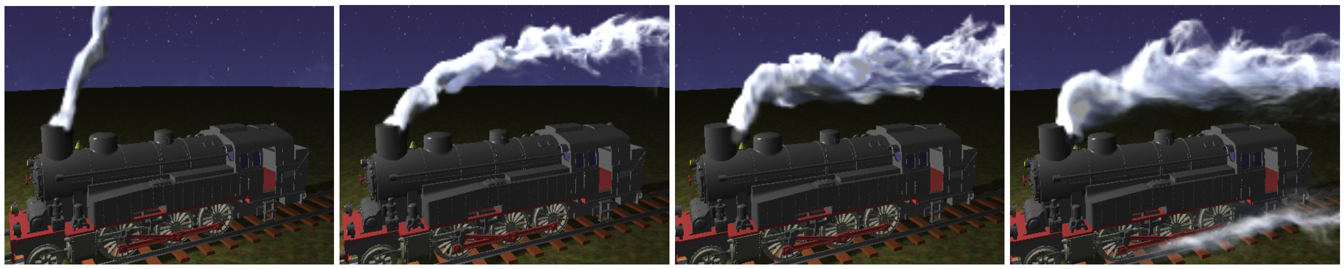

Figure 3: We apply our method to an accelerating train that, in the end, comes to an abrupt halt. Due to our energy transport model, turbulence intensities correctly adapt to the direction of the flow and the train's velocity.

and can be used to artificially increase or decrease the strength of the turbulence. This is one of the two main parameters of our model.

3.3 Anisotropy

So far we have only considered isotropic turbulence. Some important effects, however, most notably the production of turbulence, are highly anisotropic. So, neglecting anisotropy would result in turbulence structures that are not fully connected to the motion of the underlying base flow. Capturing anisotropy requires both a way to synthesize anisotropic noise, as well as an energy transport model capable of providing the anisotropic energy distribution. For the latter, Reynolds stress transport models can be used, see, e.g., [Pope 2000] for details. These models describe the evolution of tensor quantities, most notably the Reynolds stress tensor. While the complete model is far too complex to be applied in real-time applications, we will use selected elements of this theory to augment our model. According to [Pope 2000], the most relevant case of anisotropy is the production of elongated vortex structures due to shear effects. This happens, e.g., at boundaries and leads to rotational velocities perpendicular to the shear plane, reducing the dimensionality of the effect from three to two dimensions. Therefore, we consider the case of turbulence consisting of an isotropic component with energy k_{iso} that is handled as described above, and a completely anisotropic two-dimensional component with the energy vector $\mathbf{k_A}$. The direction of $\mathbf{k_A}$ defines the normal of a plane to which the anisotropic turbulence is confined. This is equivalent to a preferred rotation axis, and the magnitude of $\mathbf{k_A}$ defines the energy contained in the anisotropic vortices.

Energy Dynamics While the k–ε equations Eq. (5),(6) still hold for the total energy $k = k_{iso} + k_{an} = k_{iso} + |\mathbf{k_A}|$, we need to determine the evolution equation of the anisotropic component $\mathbf{k_A}$. The mechanisms here are similar to transport of total energy: there is a production, a dissipation and additionally, a redistribution term. Analogous to Eq. (4), the production vector is given by the turbulent viscosity ν_T and the strain. We use an eigen-decomposition to divide the strain tensor into an anisotropic component, represented with two-dimensional turbulence, and an isotropic component. In the following, λ_i denote the eigenvalues and $\mathbf{v_i}$ the eigenvectors of S_{ij}, where λ_1 has the biggest and λ_3 the smallest absolute value. Now consider the production ellipsoid defined by the vectors $\mathbf{p_i} = 2\,\nu_T\,\lambda_i^2 \mathbf{v_i}$. The plane of two-dimensional shear stress is spanned by its two longest vectors $\mathbf{p_1}$, $\mathbf{p_2}$, while the plane normal is given by $\mathbf{v_3}$. The isotropic component, on the other hand, is the sphere spanned by the smallest common component of all vectors, that is $|\mathbf{p_3}|$. Therefore, we can define the anisotropic production vector to be

$$\mathbf{P_A} = 2\,\nu_T\,(\lambda_1^2 + \lambda_2^2 - 2\,\lambda_3^2)\,\mathbf{v_3} \quad . \tag{8}$$

While in areas of high production, e.g. near obstacle boundaries, anisotropic effects can be observed, the turbulence further away from these regions is largely isotropic. This is due to the fact that transport processes lead to a quick isotropization of the turbulent flow. This is true for spatial turbulent transport as well for transport through the energy cascade. The LRR-IP model [Naot et al. 1970] models this isotropization. If we transfer this model to our turbulence setting, it yields an energy transfer rate of

$$\frac{\mathrm{D}\mathbf{k_A}}{\mathrm{D}t} = (1 - C_A)\mathbf{P_A} - C_R \frac{\varepsilon}{k}\mathbf{k_A}\,. \tag{9}$$

from $|\mathbf{k_A}|$ to k_{iso}. The standard model constants are defined as $C_A = 0.6, C_R = 1.8$. Here, the dissipation ε is generally assumed to be isotropic, as it occurs on very small scales, while the anisotropic vortices are initiated primarily on the larger scales. This means that it is sufficient to solve the isotropic Eq. (6) for dissipation.

Synthesis We extend the synthesis algorithm from § 3.2 for anisotropy by including additional turbulence bands. In these bands, two-dimensional wavelet turbulence $\omega^{\mathbf{2D}}$ is synthesized as in Eq. (7). The operator $\mathbf{R}$ rotates the noise field into a plane normal to the anisotropy vector $\mathbf{k_A}$. The total velocity $\mathbf{u}$ of a particle can therefore be determined by the equation

$$\begin{aligned}\mathbf{u} = \mathbf{U} + 2(\alpha\, k_{iso})^{\frac{1}{2}} \sum_i^N \omega_\mathbf{i}(\mathbf{q})\, 2^{-\frac{5}{6}i} + \\ 2\,|\alpha\,\mathbf{k_A}|^{\frac{1}{2}} \sum_j^M \mathbf{R}(\mathbf{k_A})\,\omega_\mathbf{j}^{\mathbf{2D}}(\mathbf{q})\, 2^{-\frac{5}{6}j}\end{aligned} \tag{10}$$

with $k_{iso} = k - |\mathbf{k_A}|$. As anisotropy decays quickly in the spectral cascade, we have found that it is sufficient to use one band of anisotropic turbulence for the largest scale.

4 Implementation

We have implemented our model to execute both the Eulerian fluid simulation and the particle based turbulence model on a GPU. For the underlying Eulerian solver, we use a typical MAC discretization with second order semi-Lagrangian advection, as described in [Bridson 2008] and [Selle et al. 2008]. Our implementation makes use of a multi-grid solver for computing the pressure correction, as described in [Cohen et al. 2010]. For the Lagrangian turbulence model, each particle stores its position and velocity as well as the turbulence parameters $k, \varepsilon, \mathbf{k_A}$ and $\mathbf{q}$. The evolution of these variables is given by integrating Eq. (5), Eq. (6), Eq. (9), and evaluating Eq. (10), respectively, on the particle system. We use a simple forward Euler integrator for all of these equations. The strain eigen-decomposition on the other hand is calculated on each grid cell. As the 3×3 strain tensor is symmetric, eigenvalues can be found efficiently using the analytic formulation [Smith 1961]. Our model is designed to work without any particle-particle interactions, and only a few linear interpolations of data from simulation grid for velocity and strain are necessary to compute the particle dynamics. This makes it very efficient to compute even in massively parallel settings. In our setup, the smoke is rendered online using half-angle volumetric shadowing by Ikits et al. [2004], enabling the complete framework to run at interactive frame-rates and therefore providing immediate results. Below, we will discuss important details concerning stability and initialization.

Stability The k–ε model, being a coupled system of two PDEs in its original form, has inherent stability problems. Especially k in the divider of Eq. (6) causes instabilities for flows with low turbulence. Therefore, the model is usually modified to guarantee stability. A commonly used approach is a low-Reynolds number treatment, as described in [Pope 2000]. We use a simplified version of this approach to ensure that a minimal turbulent energy is always present in the simulation.

A meaningful range for the turbulent energy k is given by $k = \frac{3}{2}{U_0}^2 I^2$, with the turbulent intensity $I \in [0 \dots 1]$. Here, U_0 is the characteristic velocity scale, which can be determined from the simulation parameters, e.g., the maximal speed of the car in Fig. 1. As suggested in the field of aerodynamics research [Spalart and Rumsey 2007], we use a value of $I_{min} = 10^{-3}$ as a minimal turbulent intensity, while, naturally, the maximal intensity is given by $I_{max} = 1$. By restricting the simulation to this meaningful range of values, the system quickly recovers from overshoots and is stable for arbitrary time steps.

Similarly, we can define a corresponding range for the values of ε. We obtain a minimum dissipation by specifying a minimal turbulent viscosity equal to the molecular viscosity of air ν_{air}, which represents a natural lower bound for the viscosity of smoke simulations. Inserting this into Eq. (3) yields $\varepsilon_{min} = C_\mu \frac{{k_{min}}^2}{\nu_{air}}$. The maximal dissipation, on the other hand, can be derived on the basis of a minimal turbulent length scale L_{min}, and is given by $\varepsilon_{max} = {C_\mu}^{\frac{3}{4}} {k_{max}}^{\frac{3}{2}} \frac{1}{L_{min}}$. We use a minimal length scale of $\frac{1}{10}$ of a grid cell in our simulations.

Note that these ranges for turbulent energy and dissipation are useful when allowing users to interact with the simulation. They can, e.g., provide artists with intuitive parameter ranges for setting up turbulence sources in a scene.

Initial state We seed the particles at the smoke inflow of the scene. As this will usually not coincide with the fluid inflow region, we need to specify sensible initial values for the turbulence parameters of these particles. In cases where the inlet is in a low-turbulence area, we can use the lower boundaries k_0 and ε_0 as initial values. If, on the other hand, the smoke should be generated in a turbulent region, we need to specify initial energy levels, as we have no information about the history of the particles. This can be achieved with different approaches. We estimate typical turbulent intensities for k and ε similar to the estimation of maximal bounds described in the previous paragraph, and use these values for initializing the particles. Here, the minimal length scale L_{inlet} is another important parameter of our model, and can be used to tune the amount of turbulence injected into the scene. Another possibility is to initialize particles with the lower bounds k_0 and ε_0, and then perform a small number of iterations of the turbulence model on the newly seeded particles.

Texture Advection Naturally, the structure of the turbulence should deform as given by the motion of the flow. However, using a naive approach, e.g., updating the local texture coordinate $\mathbf{q}$ of each particle using only the large scale motion with $\frac{D\mathbf{q}}{Dt} = \mathbf{u} - \mathbf{U}$ quickly destroys coherence of the turbulent structures. This would lead to uniform noise instead of recognizable swirling motions, an effect that is caused by the mixing behavior of the turbulent flow.

By construction, our aim is to update each particle without having to know about its neighbors, so it is undesirable to perform any kind of spatial interpolation on the particles.

Therefore, we rely on guiding particles to preserve the local coherence of the turbulence. Guiding particles are seeded together with the actual smoke particles, and assigned to a small group of smoke particles based on local neighborhood. On seeding, each guiding particle is assigned a fixed texture coordinate $\mathbf{q_G}$ based on its world coordinate, which acts as a local frame of reference for the texture coordinates of the attached smoke particles.

```
// Grid-based Fluid solver
Semi-Lagrangian advection of $\mathbf{U}$
Pressure projection
Calculate strain field $S_{ij}$

Seed and initialize new particles

for each particle do
    Sample $U$, $S_{ij}$ at particle position $\mathbf{x}$

    // Energy dynamics
    Compute turbulent viscosity: $\nu_T \leftarrow C_\mu \frac{k^2}{\varepsilon}$
    Compute production: $\mathbf{P} \leftarrow$Eq. (8)
    Integrate $k \leftarrow k + \Delta t \,(|\mathbf{P}| - \varepsilon)$
    Integrate $\varepsilon \leftarrow \varepsilon + \Delta t \,\frac{\varepsilon}{k}(C_1 P - C_2 \varepsilon)$
    Energy transfer: $\mathbf{k_A} \leftarrow \mathbf{k_A} + \Delta t\,(1 - C_2)\mathbf{P} - \Delta t\, C_R \frac{\varepsilon}{k} \mathbf{k_A}$
    Stabilize $k, \varepsilon$ using $k_{min,max}$ and $\varepsilon_{min,max}$

    // Motion equations
    Synthesize velocity: $\mathbf{u} \leftarrow$ Eq. (10)
    Integrate $\mathbf{x} \leftarrow \mathbf{x} + \Delta t\, \mathbf{u}$
    Integrate $\mathbf{q} \leftarrow \mathbf{q} + \Delta t\,(\mathbf{q_G} + \mathbf{x} - \mathbf{x_G})$
end for
Advect guiding particles in flow field $\mathbf{U}$

Render simulation data
```

Figure 4: Pseudo-code for the simulation loop.

As the guiding particles represent the motion of the turbulence textures, they are advected using only the large scale flow from the underlying simulation. The local texture coordinate of each smoke particle can now be computed using the local position $\mathbf{x}$ with respect to the assigned guiding particle as $\mathbf{q} = \mathbf{x} - \mathbf{x_G} + \mathbf{q_G}$. This approach allows us to efficiently preserve locality while adhering the turbulence motion to the large scale flow. While coherence and incompressibility are exactly preserved within the particle cloud of a guiding particle, coherence loss and small-scale deviations from incompressibility may appear between these clouds. Therefore, guiding particles should be seeded such that the associated clouds are compact, sized above turbulence length scale, and cover all flow paths.

While more sophisticated models for texture advection, e.g., [Yu et al. 2009], or a dynamic re-assignment of guiding particles could be used, we find that the described approach works well in practice. In our example scenes, we seed between 1 and 10 guiding particle per timestep, randomly distributed across the seeding area. We found this to be sufficient to prevent visual artifacts due to coherence loss.

The complete simulation loop is specified in the pseudo-code in Fig. 4.

5 Results and Discussion

In the following, we will discuss several simulations to highlight the features of our model and differences to previous work. We refer the reader to the accompanying video for seeing these simulations in motion.

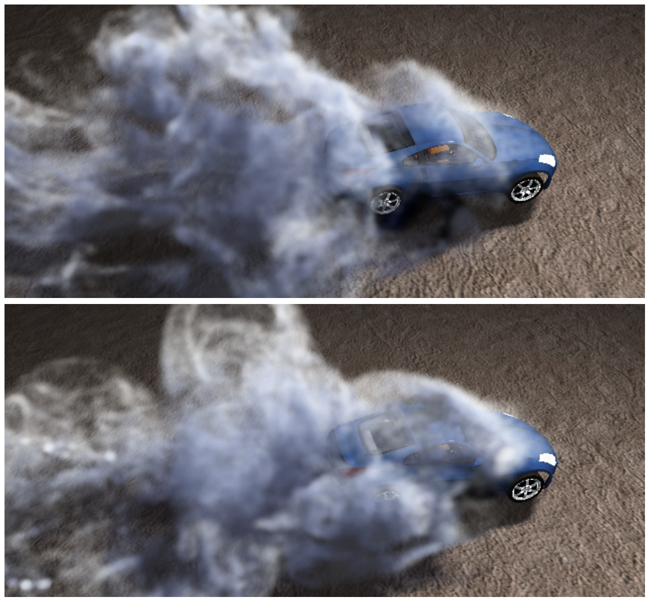
Figure 5: The wake behind a car is simulated with 1M particles. Our method (top) and the reference high-resolution solver (bottom) show similar small-scale details.

Comparison with reference simulation In order to evaluate the realism of our model, we simulate the flow in the wake of a car (Fig. 5). The simulation uses 1M particles, and a base solver resolution of $32 \times 8 \times 32$. We compare our model to a $256 \times 64 \times 256$ high-resolution reference solver. While the exact form of the turbulence is different between our method and the reference solver, we observe that both show a similar level of small-scale detail.

Energy model We demonstrate the ability of our model to handle obstacle-induced turbulence by simulating a flow over a ramp shown in Fig. 7. This setup uses a resolution of the base solver of $64 \times 16 \times 16$ grid cells. When using low grid resolutions such as this, flow instabilities induced by obstacles are dampened out, and no turbulence is induced. This effect can be seen in the top image of Fig. 7. In this example, turbulence should develop to the left of the ramp as the flow travels from right to left. Our method tracks causality in the production of turbulence, resulting in a correct swirling motion perpendicular to the edge of the step, purely behind the sharp edge. Turbulence synthesis methods such as Kim et al. [2008b] that amplify or derive turbulent energy directly from the computed velocity field do not track the causality in the production of turbulence. In this case, Wavelet turbulence incorrectly produces turbulence in the laminar region right of the edge.

In a more complex example shown in Fig. 3, we simulate a train accelerating and braking. Here, the source of turbulence is not induced by obstacles, as in the ramp example, but is due to the pulsed emission of smoke from the chimney. This is also inherently handled by the production term of our energy model. Also, correct adaptation of turbulence intensity to the train's velocity can be observed.

Anisotropy The effect of anisotropic turbulence is demonstrated in a simulation of a strongly turbulent flow past a cylinder. We seed a thin horizontal sheet of smoke to the right, visualizing only a slice of the 3D problem. The side-view of the simulation, with anisotropy handling disabled (top) and enabled (bottom) is shown in Fig. 6. If anisotropy is not handled, isotropic turbulence is injected immediately downstream of the obstacle. This leads to strong disturbances normal to the plane of motion, as can be seen in the top image of Fig. 6. Our model predicts a zone of high anisotropy behind the cylinder. Here, the turbulence is expected to be confined within the smoke sheet, therefore integrating with the large scale Karman vortices, before becoming more and more isotropic, towards the left side of the lower image.

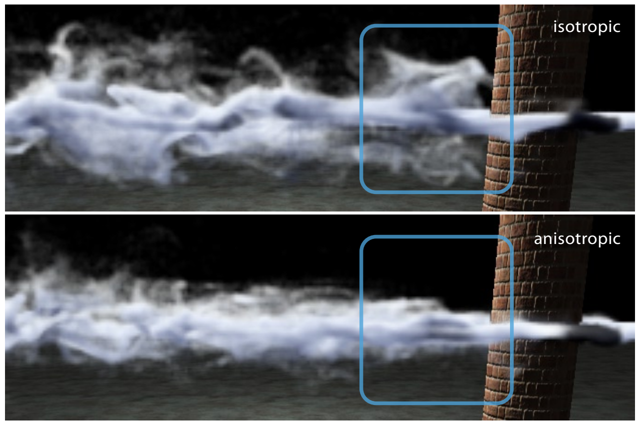

Figure 6: In this example, a thin sheet of smoke flows around a cylinder. Here, the side view is depicted. Using only the isotropic turbulence model (top), the induced turbulence disturbs the flow, as can be seen by the unrealistic spikes left of the cylinder. Using our anisotropy extensions (bottom), the turbulence integrates into the overall flow, and a smooth transition to full isotropic turbulence can be observed.

Scalability To demonstrate the scalability of our model, we simulate a smoke wake behind a car with varying particle numbers, while keeping the grid resolution fixed at $32 \times 8 \times 32$. As can be seen in Fig. 1, the large scale flow remains consistent in all cases, while the amount detail is controlled by the number of particles. As it is sufficient to use a very low grid resolution for the Eulerian solver in all examples, the performance scales approximately linearly in the number of particles. With one million particles, our model achieves 15 frame per second on average (including rendering). Increasing the number of particles to four millions, we still achieve 4.7 frames per second. The exact numbers can be found in Table 1. This means our model is able to compute accurate turbulence dynamics efficiently. GPU-based methods relying on grids are strongly limited in detail due to the available memory, the Eulerian solver of our implementation, e.g., is limited to a 128^3 resolution using the same hardware. Using our particle based approach we are, on the other hand, able to achieve very detailed motion in an efficient manner.

An example of our method in an interactive game-like setting is included in the video. The user controls a smoke emitting gun, demonstrating free stream turbulence as well as turbulence induced by obstacle interaction. Our method also opens up the possibility to compute and synthesize turbulence outside the grid-based solver. If no underlying grid is present, zero turbulence production and the last encountered large-scale velocity are taken as an input for the calculation. In the video, we show that this allows a smoke volume to leave the domain of the Eulerian simulation, while still exhibiting turbulent motion. This is very useful for interactive applications where the spatial limits of the domain should be hidden from the user.

For all of our examples, we vary only the α and L_{inflow} parameters. Recall that α controls the overall amount of turbulence and L_{inflow} controls turbulence at an inlet. Varying only these parame-

Setup	Grid res.	#part	α	L_{in}	Base [ms]	Part. [ms]	Total [fps]
Car Fig. 1	$32 \times 8 \times 32$	250k	2.5	0.04	20	7.6	34
Car	$32 \times 8 \times 32$	1M	2.5	0.04	19	27	15
Car	$32 \times 8 \times 32$	4M	2.5	0.04	20	92	4.9
Car(no turb.)	$32 \times 8 \times 32$	1M	–	–	19	6.4	20
Comp. Fig. 7	$64 \times 16 \times 16$	1M	2.6	0.08	19	23	17
Aniso. Fig. 6	$64 \times 16 \times 16$	1M	15.0	0.1	19	27	15
Iso. Fig. 6	$64 \times 16 \times 16$	1M	15.0	0.1	14	27	16
Smoke gun	$48 \times 48 \times 48$	1M	4.2	0.02	25	18	18
Train Fig. 3	$64 \times 32 \times 16$	6M	3.0	0.05	44	161	3.7

Table 1: Performance numbers for our simulation runs. Timings are given per frame. *Base* refers to the grid-based solver, while *Part.* represents turbulence computation, synthesis and particle system update. The total framerate includes both simulation and online rendering. All simulations were run on a NVidia GTX 480 graphics card on a workstation with an Intel Core i7 CPU and 8GB of RAM.

ters allows for artistic control while retaining visual realism.

6 Conclusions

We have presented a novel scalable algorithm for simulating anisotropic turbulence. By separating the system into a grid-based solver and a decoupled particle system without particle-particle interactions, our method is highly efficient on parallel systems. The algorithm is driven by an anisotropic energy transport mechanism, and handles both free stream turbulence production and turbulence induced by walls. Turbulence is synthesized directly on the rendered particles, which allows the simulation to handle the full detail that will later on be displayed, while not wasting any processor cycles for regions that are not visible. This way, we achieve frame rates of more than 15 frames per second even for detailed simulations with millions of particles.

On the other hand, our algorithm can exhibit artifacts when the underlying simulation is not able to resolve all features of a flow, e.g., in the presence of very thin objects. Analytic wall functions, e.g., like those in [Pfaff et al. 2009], could help to provide a more accurate turbulence seeding with a coarse flow representation. As our underlying coarse grid solver [Cohen et al. 2010] can only handle first-order accurate boundary conditions, stair-step artifacts may appear around solid curved obstacles. It would therefore be interesting to pair our method with a more accurate real-time solver, and to extend our particle based simulation to handle sub-grid geometric detail.

Another limitation of our approach is the restriction to single phase fluid simulations. Also, the algorithm does not perform well for large, non-turbulent smoke volumes, which can have unnecessarily large numbers of particles inside the volume that hardly move. Both of these points are interesting venues for future research. It would be highly interesting to extend our model to free surface flows for liquids, and use an adaptive particle representation to handle larger smoke volumes more efficiently.

In addition, we plan to extend our framework to automatically adapt the level-of-detail for large interactive scenes, as the modularity of our approach makes it highly suitable to combine different simulation approaches. This will allow us to smoothly transition from a simple static flow field, to a Eulerian fluid simulation, while finally adding detail with our anisotropic Lagrangian turbulence model.

Acknowledgments The authors would like to thank the SIGGRAPH Asia reviewers for their comment and suggestions, and everyone at CGL for the valuable discussions. Tobias Pfaff is supported by the ETH grant *TH-23 07-3*.

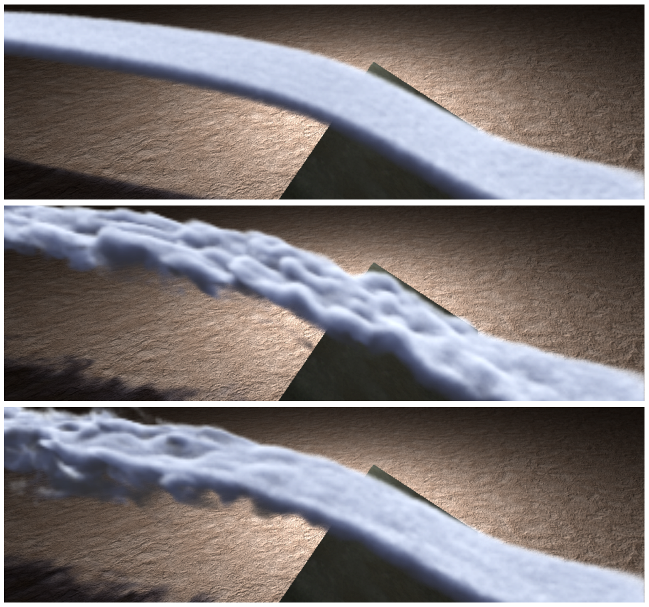
Figure 7: A flow over a ramp is simulated. The low-resolution solver (top) does not represent the flow instability after the edge of the ramp. Therefore methods like Wavelet Turbulence (middle) that depend on the solver for energy calculations also fail to catch the correct turbulence seeding region. Our model (bottom) is able to predict turbulence production due to a full energy transport model.

References

BRIDSON, R., HOURIHAM, J., AND NORDENSTAM, M. 2007. Curl-noise for procedural fluid flow. *ACM SIGGRAPH papers 26*, 3, Article 46.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. A K Peters.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 377–384.

COHEN, J., TARIQ, S., AND GREEN, S. 2010. Interactive fluid-particle simulation using translating eulerian grids. In *Proceedings of the 2010 SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

COOK, R., AND DEROSE, T. 2005. Wavelet noise. In *Proceedings of ACM SIGGRAPH 2005*, vol. 25.

CRANE, K., TARIQ, S., AND LLAMAS, I. 2007. *GPU Gems 3*. ch. Real-time Simulation and Rendering of 3D Fluids.

ENRIGHT, D., FEDKIW, R., FERZIGER, J., AND MITCHELL, I. 2002. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 83–116.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. Animating gases with hybrid meshes. In *Proceedings of ACM SIGGRAPH*.

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

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004) 23*, 3, 463–468.

HONG, J., AND KIM, C. 2003. Animation of bubbles in liquid. *Proceedings of Eurographics 2003 22*, 3.

HORVATH, C., AND GEIGER, W. 2009. Directable, high-resolution simulation of fire on the gpu. *ACM SIGGRAPH papers*.

IKITS, M., KNISS, J., LEFOHN, A., AND HANSON, C. 2004. *GPU Gems: Programming techniques for real-time Graphics*.

IRVING, G., GUENDELMAN, E., LOSASSO, F., AND FEDKIW, R. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. (SIGGRAPH Proc.) 25*, 3, 805–811.

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

KIM, D., YOUNG SONG, O., AND KO, H.-S. 2008. A semi-lagrangian cip fluid solver without dimensional splitting. *Comput. Graph. Forum (Proc. Eurographics) 27*, 2, 467–475.

KIM, T., THUEREY, N., JAMES, D., AND GROSS, M. 2008. Wavelet turbulence for fluid simulation. *ACM SIGGRAPH Papers 27*, 3 (Aug), Article 6.

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

LAUNDER, B. E., AND SHARMA, D. B. 1974. Applications of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Lett. Heat Mass Transf. 1*, 1031–138.

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

MOLEMAKER, J., COHEN, J. M., PATEL, S., AND NOH, J. 2008. Low viscosity flow simulations for animation. In *ACM SIGGRAPH / Eurographics Symp. on Comp. Anim.*, 9–18.

MULLEN, P., CRANE, K., PAVLOV, D., TONG, Y., AND DESBRUN, M. 2009. Energy-Preserving Integrators for Fluid Animation. *ACM SIGGRAPH Papers 28*, 3 (Aug), Article 38.

MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. 2005. Particle-based fluid-fluid interaction. *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*.

NAOT, D., SHAVIT, A., AND WOLFSHTEIN, M. 1970. Interaction between components of the turbulent velocity correlation tensor due to pressure fluctuations. *Israel J. Technol. 8*, 259–269.

NARAIN, R., SEWALL, J., CARLSON, M., AND LIN, M. C. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM SIGGRAPH Asia papers*, Article 166.

OBUKHOV, A. 1941. The spectral energy distribution in a turbulent flow. *Dokl. Akad. Nauk 32*, 22–24.

PANCHEV, S. 1971. *Random Functions and Turbulence*. Oxford: Pergamon Press.

PFAFF, T., THUEREY, N., SELLE, A., AND GROSS, M. 2009. Synthetic turbulence using artificial boundary layers. *ACM Transactions on Graphics 28*, 5, 121:1–121:10.

POPE, S. B. 2000. *Turbulent Flows*. Cambridge University Press.

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

SCHECHTER, H., AND BRIDSON, R. 2008. Evolving sub-grid turbulence for smoke animation. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation*.

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*.

SMITH, O. K. 1961. Eigenvalues of a symmetric 3×3 matrix. *Comm. of the ACM 4*.

SPALART, P. R., AND RUMSEY, C. L. 2007. Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA Journal 45*, 10.

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*.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular Bases for Fluid Dynamics. *ACM SIGGRAPH Papers 28* (Aug), Article 39.

WILCOX, D. C. 1993. *Turbulence modelling for CFD*. DCW Industries.

YU, Q., NEYRET, F., BRUNETON, E., AND HOLZSCHUCH, N. 2009. Scalable real-time animation of rivers. *Comput. Graph. Forum 28*, 2, 239–248.

ZHAO, Y., YUAN, Z., AND CHEN, F. 2010. Enhancing fluid animation with adaptive, controllable and intermittent turbulence. *ACM Eurographics*.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as fluid. *ACM SIGGRAPH papers*.