Synthetic Turbulence using Artificial Boundary Layers

Tobias Pfaff[1] Nils Thuerey[1] Andrew Selle[2] Markus Gross[1,3]

[1] Computer Graphics Lab, ETH Zurich [2] Walt Disney Animation Studios [3] Disney Research Zurich

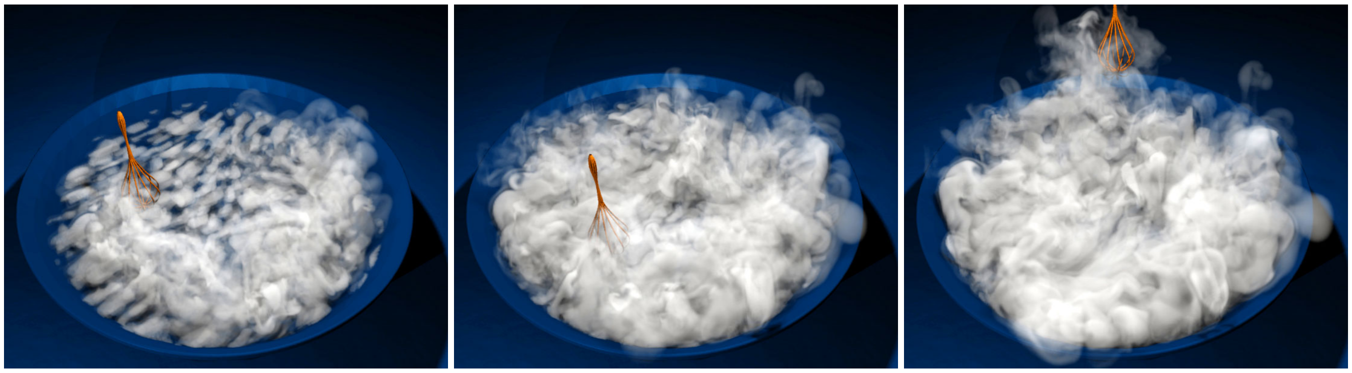

Figure 1: Our algorithm allows us to precompute detailed boundary layer data and efficiently reuse it for new simulations. We are able to generates turbulent vortices taking into account the relative velocity of an obstacle in the flow. Here, we apply our algorithm to a very thin object that is barely represented on the simulation grid.

Abstract

Turbulent vortices in fluid flows are crucial for a visually interesting appearance. Although there has been a significant amount of work on turbulence in graphics recently, these algorithms rely on the underlying simulation to resolve the flow around objects. We build upon work from classical fluid mechanics to design an algorithm that allows us to accurately precompute the turbulence being generated around an object immersed in a flow. This is made possible by modeling turbulence formation based on an averaged flow field, and relying on universal laws describing the flow near a wall. We precompute the confined vorticity in the boundary layer around an object, and simulate the boundary layer separation during a fluid simulation. Then, a turbulence model is used to identify areas where this separated layer will transition into actual turbulence. We sample these regions with vortex particles, and simulate the further dynamics of the vortices based on these particles. We will show how our method complements previous work on synthetic turbulence, and yields physically plausible results. In addition, we demonstrate that our method can efficiently compute turbulent flows around a variety of objects including cars, whisks, as well as boulders in a river flow. We can even apply our model to precomputed static flow fields, yielding turbulent dynamics without a costly simulation.

Keywords: Turbulence, Physically Based Animation, Fluid Simulation

ACM Reference Format
Pfaff, T., Thuerey, N., Selle, A., Gross, M. 2009. Synthetic Turbulence using Artificial Boundary Layers. *ACM Trans. Graph. 28*, 5, Article 121 (December 2009), 10 pages. DOI = 10.1145/1618452.1618467 http://doi.acm.org/10.1145/1618452.1618467.

1 Introduction

Many interesting natural phenomena are the result of fluids interacting with objects immersed in a flow. Examples include smoke behind a car being influenced by the vortices in the car's wake as well as complex wave structures and splashes forming around rocks in river beds. Fluid simulations have made huge steps in previous years, and have become an important tool for filmmakers. Despite the increased popularity, fluid simulations are still inherently difficult to use. For a typical solver, the accuracy of the spatial discretizations and the viscosity of the fluid to be simulated are related. This means that a fluid such as air or water requires a high grid resolution to yield a turbulent and interesting look. This unfortunately results in long computation times and high memory requirements, limiting controllability and the scope of applications.

Although several approaches to alleviate these fundamental problems were suggested, most of them deal with the preservation of vortices already represented in the overall flow. In contrast to this, our work targets the interaction of objects with the fluid, namely turbulence created by flows around obstacles. While previously used methods either require a manual seeding of turbulence [Selle et al. 2005] or rely on the simulation to resolve the interaction with the object accurately enough [Kim et al. 2008b; Narain et al. 2008], we precompute the turbulence generated around an object using techniques from the wall flow theory of traditional computational fluid dynamics (CFD). Such theory models the behavior of near-wall regions of flow around objects, which is important since these boundary layers strongly influence forces on the object as well as shed turbulent structures. These techniques have been important for many applications such as the optimization of wing profiles.

Our precomputation step captures the characteristics of the boundary layer around the object, and stores it for different sets of flow directions. Our model allows us to purely resolve the geometry of the object with the precomputation, instead of having to fully resolve the actual flow velocity in the often very thin boundary layer. For the precomputation, we assume that an object can be characterized by a relative translational and rotational velocity, allowing for simulations of rigid body motion or static flows of arbitrary direction.

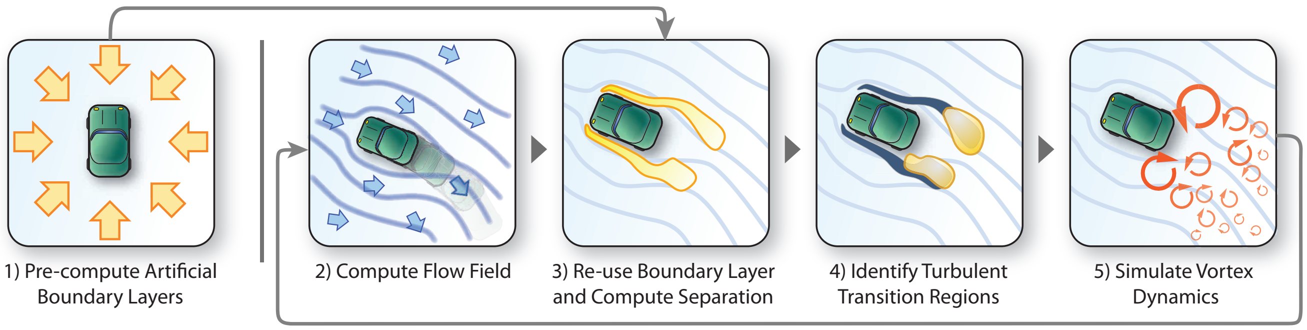
1) Pre-compute Artificial Boundary Layers
2) Compute Flow Field
3) Re-use Boundary Layer and Compute Separation
4) Identify Turbulent Transition Regions
5) Simulate Vortex Dynamics

Figure 2: An overview of different steps of our algorithm. After precomputing the artificial boundary layer (1), we run a new simulation (2) and apply the confined vorticity from the precomputation (3). Regions transitioning into turbulence are identified with an approximation of the Reynolds stress (4). This results in the creation of vortex particles. Their dynamics are computed in an additional step (5).

In a second step, we re-use our precomputed boundary layer data to efficiently calculate where vortices are created around the object in a separate simulation. We compute the evolution of vorticity around the object, and estimate regions where this vorticity becomes unstable to form actual turbulent vortices. In contrast to traditional CFD applications, we are not targeting averaged properties, such as the mean pressure distribution around an object. Our goal is to compute a visually interesting temporal evolution of the turbulence around the object. We achieve this by modeling the vortices with an improved variant of vortex particles [Selle et al. 2005]. We create vortex particles based on boundary layer vorticity and simulate their dynamics (as well as splitting and merging) based on turbulent energy transport and the vorticity equations of flow. The resulting vorticity can even be reconstructed into a grid with higher resolution than the base simulation.

Overall, our approach allows us to efficiently compute complex turbulent flows around objects without having to manually seed particles or having to simulate on very high grid resolutions. Moving the computations for the generation of turbulence into a preprocessing step allows us to quickly set up new simulations around a given object. The contributions of our method are:

- A new method to accurately track and precompute boundary layer vorticity using an *artificial boundary layer* whose accuracy is independent of a final simulation resolution.
- An algorithm to process the precomputed data in a dynamic simulation to spawn vortices according to the current flow.
- A vortex particle method that models vortex interactions and adheres to turbulent energy transport theory.

Our method also elegantly handles turbulent free surface flows, a topic that has been barely studied in previous work other than the work of Selle et al. [2005] and Narain et al. [2008]. In addition, our method can accurately compute turbulent wakes around objects that are too fine to be resolved on the simulation grid. This is due to the fact that we decouple the resolution, and thus accuracy, of computations for the turbulence around obstacles, the actual simulation and the evaluation of the vortex particles. Because our method separates resolving the boundary layer and the actual flow, it can be seen as an example of the increasingly important paradigm of *multi-scale physics*.

Overview Our simulation method consists of a standard, grid-based fluid solver, e.g. according to Stam [1999], augmented with a turbulence representation. We use an enhanced version of vortex particles [Selle et al. 2005], which enforce a rotation in the flow around the particle's position, to represent turbulence. The key point of our paper is an intelligent method to seed these particles. In § 3, we will develop a theory for this. In § 4, the dynamics of the vortex particles are described, and the coupling of the vortex particles to the flow field is explained. The actual simulation loop and implementation details are discussed in § 5.

2 Related Work

Stam [1999] popularized the unconditionally stable combination of semi-Lagrangian advection with first order pressure projection, but numerical dissipation still plagued simulations. A common way to combat dissipation is to attempt to more closely converge to the solution. Adaptive grid methods [Losasso et al. 2004], [Feldman et al. 2005], [Irving et al. 2006] address numerical dissipation with refinement. However, this assumes only a small part of a simulation needs high detail. One can also apply higher order advection methods such as Back and Forth Error Compensation and Correction [Kim et al. 2005], semi-Lagrangian MacCormack [Selle et al. 2008], QUICK [Molemaker et al. 2008], or CIP methods (see e.g. [Kim et al. 2008a]). Alternatively, Fedkiw et al. [2001] advocate detecting and amplifying existing vortices to combat dissipation. Mullen et al. [2009], on the other hand, propose an implicit energy preserving velocity integration scheme to solve the vorticity equation. Although precomputations are difficult for fluid simulations, Wicke et al. [Wicke et al. 2009] presented a method to precompute and couple reduced bases of flows. These algorithms, however, still rely on the underlying simulation to resolve the boundary layers around objects.

Many researchers have augmented or replaced basic fluid simulation with synthetic turbulence. Stam [1993] introduced a method that used a Kolmogorov spectrum to produce procedural divergence free turbulence. This approach was used to model nuclear explosions and flames [Rasmussen et al. 2003; Lamorlette and Foster 2002]. Bridson et al. [2007] suggest taking the curl of vector noise fields to produce divergence free velocity fields. They explicitly address computing flows around objects efficiently by modulating the potential field, however, their method does not model the complex turbulent effects near the wall. Most recently, researchers have considered transporting quantities with a lower resolution simulation which is used to generate detail [Kim et al. 2008b; Schechter and Bridson 2008; Narain et al. 2008]. However, the treatment of obstacles is not the focus of these techniques. Kim et al. [2008b] only extrapolate energies into obstacles to prevent artifacts and Narain et al. [2008] state that their approximation is not valid in the vicinity of obstacles.

While the preceding approaches achieve interesting turbulent behavior in fluid volumes, a major source of turbulence is interaction of fluids with solids. Two-way coupling of fluids has been addressed by Carlson et al. [2004], Guendelmann et al. [2003] and Klingner [2006]. More recently researchers have modeled subgrid interactions with objects more accurately through the use of apertures [Batty et al. 2007; Robinson-Mosher et al. 2008]. Nevertheless, very little previous work in graphics addresses the problem that the thin turbulent boundary layer is not resolved in the simulation, resulting in turbulence not being shed.

3 Wall-Induced Turbulence

While turbulence in flows is generated by various processes, the most common and visually important one is turbulence generation at the flow boundaries. Therefore, our algorithms explicitly model this important process. Our method is based on wall flow theory and turbulence modeling, which we will outline in § 3.1. A more thorough review can be found in the book by Pope [2000].

3.1 Generation of turbulence

In wall-bounded flows, wall friction enforces a tangential flow velocity of zero at the wall. This leads to the formation of a thin layer with reduced flow speed, called the boundary layer. Fig. 3 shows a velocity profile in the boundary layer. It has been shown that this profile is equivalent for all wall-bound flows when using normalized units. This *universal law of the wall* was stated by van Driest in [1956].

The gradient of tangential flow velocity in the boundary layer leads to the creation of a thin sheet of vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{u}$. For planar walls, this vorticity remains mostly confined to the boundary layer, and we will thus refer to it as *confined vorticity*. At regions of high flow instability however, vorticity may be ejected from the boundary layer and enter the flow as turbulence. This happens e.g. at sharp edges, where the boundary layer is separated from the wall, and likely to become unstable, or when other turbulent structures disturb the boundary layer. This process of turbulence formation is referred to as roll-up, and is the predominant mechanism of wall-induced turbulence generation [Jiménez and Orland 1993]. There is no theory quantitatively describing the boundary layer roll-up process. We will therefore model this process in a statistical sense, as explained below.

Turbulence modeling We base our approach on CFD turbulence modeling techniques. These techniques model statistical properties of turbulence based on the ensemble-average of the flow field $\mathbf{u}$. For a quasi-static flow, ergodicy permits us to use the time-averaged flow field, i.e. the flow field with all fluctuating turbulent structures averaged out, instead of the more complicated ensemble averaging.

One of the most important quantities that can be modeled in such a way is the Reynolds stress tensor. It governs the transfer of energy from the bulk flow to turbulent structures, and thus the generation of turbulence. This fact is commonly used for Reynolds-averaged Navier-Stokes simulations (RANS) or Large-Eddy Simulations (LES), and we will make use of it for our method as well. Next, we will describe how we model the boundary layer.

Boundary layer modeling In order to accurately model wall-induced turbulence formation, we need to track the confined vorticity, simulate the boundary layer separation and finally identify the transition points to turbulence.

As the boundary layer, attached to an obstacle is very thin (smaller than simulation grid resolution in most cases), it is difficult to directly measure confined vorticity. Instead, we leverage the universal law of the wall, and note that the confined vorticity only depends on the velocity scale and materials constants. For each point in the wall-attached boundary layer we therefore determine the confined vorticity as

$$\boldsymbol{\omega}_{ABL} = \beta(\mathbf{U}_s \times \mathbf{n}) \ . \tag{1}$$

The velocity scale $\mathbf{U}_s$ is the tangential component of the averaged flow velocity just outside the boundary layer. The constant β accounts for the two material constants, skin friction coefficient and the fluid viscosity. In our model, β is a user-defined parameter. We call the resulting field $\boldsymbol{\omega}_{ABL}$ the *artificial boundary layer*.

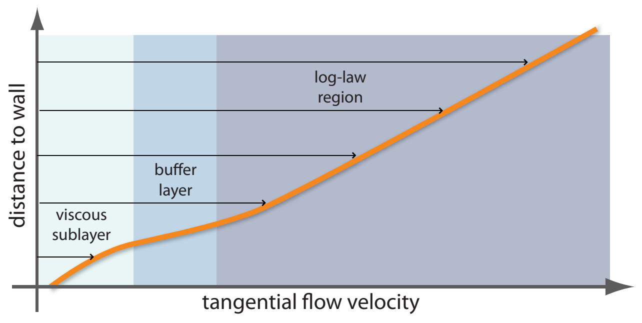

Figure 3: The mean velocity profile near a wall (in normalized units) has the form shown above. This has been confirmed in numerous experiments, and was formulated as a universal law by van Driest.

On the other hand, boundary layer separation is an advective transport process. If the wall-attached part of the artificial boundary layer is known, then the separation plume can be derived by advecting this field with the flow field during the simulation run.

The last missing part is to identify regions where the separated boundary layer becomes unstable, and the confined vorticity $\boldsymbol{\omega}_{ABL}$ transitions to free turbulence. The anisotropic part of the Reynolds tensor a_{ij}, which is responsible for the production of turbulence, is a good indicator for such transition regions. We therefore define a transition probability density p_T, which is used to seed turbulence,

$$p_T = c_P \Delta t \frac{\|a_{ij}\|}{|\mathbf{U_0}|^2} \tag{2}$$

such that regions with high Reynolds stresses are likely transition regions. Here, $\|\cdot\|$ denotes the Euclidean matrix norm. Reynolds stresses are normalized to a uniform scale by the inflow velocity $\mathbf{U}_0$, and c_P is a parameter to control the seeding granularity. If using varying time-steps, p_T has to be multiplied by Δt to ensure consistent behavior. In the following, we will describe how to compute the Reynolds stress tensor based on stresses in the averaged flow field.

Reynolds models The anisotropic component a_{ij} of the Reynolds stress tensor R_{ij} can be expressed using the turbulent viscosity hypothesis

$$a_{ij} = -2\nu_T S_{ij} \ , \tag{3}$$

where ν_T is the turbulent viscosity and S_{ij} denotes the strain tensor

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \ . \tag{4}$$

The turbulent viscosity can be expressed in terms of a so called *mixing length* l_m, which in near-wall regions is given by the distance to the wall. We chose the model of Baldwin [1978] for modeling the turbulent viscosity:

$$\nu_T \approx {l_m}^2 \|\Omega_{ij}\| \ . \tag{5}$$

Here, Ω_{ij} is the rotation tensor that can be computed as

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right) . \tag{6}$$

Using these standard methods, it is possible to model the generation of turbulence using only the non-turbulent mean flow velocities.

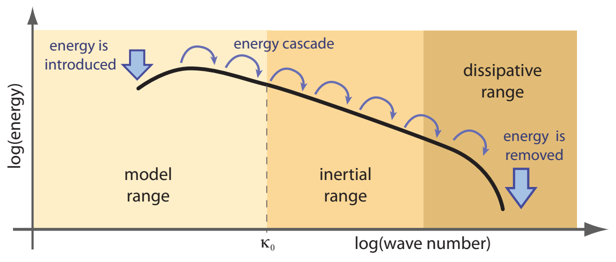

Figure 4: This graph shows the typical evolution of the energy per vortex wavenumber. Energy is introduced into the system at large scales in the model range. The energy is subsequently transferred into smaller scales by scattering of vortices, and finally dissipates due to viscosity in the dissipative range.

However, the presented Reynolds stress model requires a high grid resolution around the boundaries to capture the thin boundary layer accurately. In a typical fluid simulation in graphics, the boundary layer thickness is often smaller than a grid cell. Consequently, the discrete S and Ω operators will fail to capture the desired effect, or even cause instabilities due to highly discontinuous gradients, as also mentioned by, e.g. Narain [2008].

We therefore propose two changes to this model. First, we know that in regimes close to a wall, the norm of the rotation tensor equals the norm of the confined boundary layer vorticity, $\|\Omega_{ij}\| = |\boldsymbol{\omega}_{ABL}|$. Also, we assume that $\|S_{ij}\| \approx \|\Omega_{ij}\|$. This is a good approximation if the velocity gradient is dominated by the component normal to the wall [Pope 2000], which, except for sharp corners, is usually the case in the near-wall region. With these assumptions, we can rewrite the Reynolds stress without the problematic discrete stress and rotation tensors as

$$\|a_{ij}\| \approx 2{l_m}^2 |\boldsymbol{\omega}_{ABL}|^2. \tag{7}$$

Combined with Eq. (2) this leads to the final equation for the transition probability.

$$p_T = 2\,c_P\,\Delta t\,{l_m}^2\,\frac{|\boldsymbol{\omega}_{ABL}|^2}{|\mathbf{U}_0|^2}\;. \tag{8}$$

The seeding process for vortex particles, based on p_T, is explained in § 4.3.

3.2 Precomputing the Artificial Boundary Layer

The artificial boundary layer together with Eq. (8) can be used to seed turbulence, in the form of vortex particles, in the appropriate places of the flow. However, the expression for the wall-attached $\boldsymbol{\omega}_{ABL}$ depends on the averaged flow field $\mathbf{U}$, which is not accessible during the simulation. It is not possible to use the instantaneous flow field of the simulation, as the emerging turbulence would lead to feedback loops. However, we can precompute $\boldsymbol{\omega}_{ABL}$ for quasi-static scenes or scenes with rigidly moving objects. This has the additional advantage that we can choose simulation resolution and precomputation resolution independently, allowing us to precompute fine boundary geometries, while running the simulation on a coarse grid.

Precomputation is done by running a standard fluid solver, and time-averaging the flow field. At all obstacle boundary voxels, Eq. (1) is evaluated, and $\boldsymbol{\omega}_{ABL}$ is stored in a suitable data structure (see pseudo-code Fig. 5). More details on the implementation of the precomputation step, and how the precomputed data is used in the simulation will be given in § 5. In the next section, we will explain how to compute the dynamics of our turbulence representation.

```
1: Perform standard grid-based simulation
2: Obtain time-averaged flow field $\mathbf{U}$
3: for each voxel $\mathbf{x}$ on the obstacle boundary do
4:     // Get voxel outside the boundary layer
5:     $\mathbf{x}_e \leftarrow \mathbf{x} + l\,\mathbf{n}$
6:     $\boldsymbol{\omega}_{PRE} \leftarrow \beta\,(\mathbf{U}(\mathbf{x}_e) \times \mathbf{n})$
7:     Store $(\mathbf{x}, \boldsymbol{\omega}_{PRE})$ in a point set
8: end for
```

Figure 5: Pseudo-code for precomputing the Artificial Boundary Layer. **n** denotes the surface normal and l is the boundary layer thickness. l is chosen to be the distance from the wall at which the velocity gradient approaches zero, usually 1-2 grid cells.

4 Turbulence Synthesis

We chose to represent turbulence using vortex particles. Particles have the advantage that advection is trivial in the Lagrangian setting, while Eulerian approaches have to combat artifacts due to strong deformations from the advection [Kim et al. 2008b]. In addition, particles allow us to focus on sampling the regions where turbulence is actually generated. Narain et al. [2008] use particles with curl noise textures as a turbulence representation. However, this only works in the inertial subrange. Since we also want to simulate the model-dependent range, where no uniform direction and energy distribution can be assumed, we use an enhanced variant of the method by Selle [2005]. In contrast to the original paper, we also model energy transfer and make use of an improved synthesis step.

4.1 Energy spectrum

The statistical properties of turbulent kinetic energy can be described by energy distribution spectra. Fig. 4 shows a typical logarithmic spectrum of energy over the wavenumber of the turbulent structures. Three sections can be identified:

- *Model-dependent range*. In this region, large-scale structures are dominant and most of the spectrum's energy is contained. The production of turbulence mainly occurs in the model-dependent range, and its behavior is strongly dependent on the flow, and is therefore not easily described by statistics.
- *Inertial subrange*. The inertial subrange is a range of fully-developed homogeneous turbulence, with a behavior that is essentially equivalent for all flows. Kolmogorov [1941] found that in fact the slope of the energy spectrum is always $-5/3$. Turbulent energy in the inertial subrange flows from small to higher wavenumbers, creating an energy cascade.
- *Dissipative range*. The main energy dissipation occurs in the range of large wavenumbers.

Most existing turbulence methods in graphics are only able to model the turbulent behavior in the inertial subrange, e.g. by curl noise (see [Bridson et al. 2007; Kim et al. 2008b; Narain et al. 2008]), where a uniform distribution of rotation angles and energy can be expected. Therefore the grid resolution has to be high enough to cover all structures up to κ_0 in Fig. 4. We propose a method that also handles behavior in the model-dependent range, thus producing higher detail while requiring less simulation resolution.

4.2 Vortex particle dynamics

Turbulence dynamics can be seen from two points of view: The vorticity differential equation describes the direct evolution of the vorticity field, while the energy transport equation describes its statistical behavior. Both consist of terms for advection, generation, dissipation and scale transfer, but have different advantages for a Lagrangian representation. While the vorticity equation is well suited for describing dynamics, the injection and dissipation of energy via particle creation and dissipation is easier in an energy formulation. We will use a combination of both representations to leverage the strengths of both models.

Motion equation The dynamics of vorticity represented are described by the vorticity formulation of the Navier-Stokes equation

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \mu \nabla^2 \boldsymbol{\omega} + \nabla \times \mathbf{f}\,, \tag{9}$$

where $\mathbf{f}$ denotes the external forces, and μ is the viscosity coefficient. The left side of the equation is inherently handled by advecting the particles in the flow field of the Navier-Stokes solver. The first term on the right-hand side is the vortex stretching term. It is computed by trilinear interpolation of the discrete gradient of the velocity grid, and is used to adjust the particles' vorticity magnitude by $\Delta t(\boldsymbol{\omega} \cdot \nabla)\mathbf{u}$. This term is problematic as it might introduce exponential accumulation of vorticity magnitude in a particle. Therefore, the particle is rescaled after the update to preserve the magnitude, effectively only spinning the particle, but not altering its strength. The energy gain and loss is handled explicitly by energy dynamics, explained below. Similarly, the viscous diffusion term (the second term on the right hand side of Eq. (9)), will be handled by energy dynamics, as it is not well represented by a vorticity exchange among sparsely sampled particles.

The last term of Eq. (9) is handled with an underlying simulation, as external forces such as gravity are well-represented by a Navier-Stokes solver, and act on the vortex particles via the velocity grid. This gives us a reduced formulation of Eq. (9) that conserves vorticity as well as energy. It is therefore orthogonal to the energy transfer, the computation of which we will describe next.

Energy dynamics The dynamics of the turbulent kinetic energy E are covered by the energy transport equation [Pope 2000]

$$\frac{\partial E}{\partial t} + (\mathbf{u} \cdot \nabla)E = -\nabla \cdot \mathbf{T} + \mathscr{P} - \varepsilon \tag{10}$$

The left hand side again is handled by advection of the particles. The right-hand side consists of production $\mathscr{P}$, dissipation ε and the energy transfer term $\nabla \cdot \mathbf{T}$. These quantities are in general complex to model, especially in the model-dependent range. For the production term, we can use the information from our artificial boundary layer (see § 3.2). The dissipation ε occurs at wavenumbers that are usually well below the resolved grid resolutions. Dissipation is therefore implemented by removing particles whose radii are too small to be represented on the grid. We use a threshold of $2\Delta x$ for our simulations. Finally, for handling the remaining energy transfer term for $\mathbf{T}$ of Eq. (10), we distinguish the following two cases:

1. *The particle is in the inertial subrange*. We represent the energy cascade by decaying a particle with wavenumber κ_a into n particles of smaller wavenumber κ_b. We typically use n=2 in our simulations. From Kolmogorov's law, we can derive a timescale of decay as $\Delta t = C(\kappa_a^{-\frac{2}{3}} - \kappa_b^{-\frac{2}{3}})$, where C denotes a parameter that depends on the rate of dissipation ε. In practice we can use a value normalized by the averaged flow $\mathbf{U}$ here. We also know that for the turbulent energies $E_a/E_b = n\,(\kappa_a/\kappa_b)^{-\frac{5}{3}}$ holds, which is used to derive the vorticity magnitude of the new particles. For practical reasons, we also add a small position and angle displacement to the new vortex particles, as they would otherwise lump together.

2. *The particle is in the model-dependent range*. As transfer cannot be easily described in this regime, a simple heuristic is used. Typically, small vortices with aligned direction tend to form larger vortices in this range. Therefore, we merge vortex particles in the model-dependent range with a distance of less than the particle radius to a single larger vortex particle. Here, the vortex magnitude is chosen so that total the energy is conserved as $E_{new} = E_1 + E_2$. As very small and strong vortex particles might induce stability problems, we also conserve the energy density, i.e. $\frac{E_{new}}{V_{new}} = \frac{E_1}{V_1} + \frac{E_2}{V_2}$. This specifies the radius and strength of the merged particle. The new direction is obtained by a weighted average, with the respective energies as a weight.

4.3 Vorticity Synthesis

Turbulence synthesis is performed by enforcing the vortex particles' vorticity on the simulation velocity grid. Each vortex particle has a vorticity vector $\boldsymbol{\omega}_P$, encoding magnitude and rotation axis, and a kernel over which this value is applied.

Kernel For the direct regulation of vorticity, a kernel with the following properties is desired: at the vortex particle center, vorticity should be equal to $\boldsymbol{\omega}_P$. Also, the resulting vector field should mainly contain rotation around $\boldsymbol{\omega}_P$, and smoothly fade out with the particle radius without causing discontinuities. And lastly, the associated velocity field, and its integral, which is needed for e.g. energy calculation, should be a simple analytic form. We chose a Gaussian peak with standard deviation of σ in the vector potential to meet these requirements. In cylindrical coordinates, it is given by

$$\Psi(z,\varphi,\rho) = -|\boldsymbol{\omega}_P|\,\sigma^2 \exp^{\frac{-\rho^2-z^2}{2\sigma^2}}\, \mathbf{e}_z\,, \tag{11}$$

where the vortex axis $\mathbf{e}_z$ is aligned with $\boldsymbol{\omega}_P$. We can then derive the velocity field

$$\mathbf{u} = \nabla \times \Psi = -|\boldsymbol{\omega}_P|\,\rho \exp^{\frac{-\rho^2-z^2}{2\sigma^2}}\, \mathbf{e}_\varphi\,, \tag{12}$$

and the vorticity kernel

$$\boldsymbol{\omega} = \nabla \times \mathbf{u} = -\frac{|\boldsymbol{\omega}_P|}{\sigma^2}\left(\rho z \mathbf{e}_\varphi - (\rho^2 - 2\sigma^2)\mathbf{e}_z\right) \exp^{\frac{-\rho^2-z^2}{2\sigma^2}}\,. \tag{13}$$

A cut-off radius is used to make the kernel support finite. We use $r = \sqrt{6}\sigma$ at which point the exponential term of the kernel function has fallen to 10^{-3}. The length scale is defined at the kernels' origin, so that its wavenumber is $\kappa = \frac{1}{\sigma}$. For the contained energy $E \propto {\boldsymbol{\omega}_P}^2\,\sigma^5$ holds.

Kernel Evaluation This is a three-step process: First, the vorticity field $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ of the velocity grid is computed by finite differences. Second, all particle vorticity kernels are summed up to obtain a desired vorticity field $\boldsymbol{\omega}_D$. And third, each particle adds its kernel to the velocity field, scaled by a weight w_k, computed as:

$$w_k = \frac{\sum_{kernel}(\boldsymbol{\omega}_D - \boldsymbol{\omega}) \cdot \bar{\boldsymbol{\omega}}_P}{\sum_{kernel} \boldsymbol{\omega}_D \cdot \bar{\boldsymbol{\omega}}_P}\,, \tag{14}$$

where $\bar{\boldsymbol{\omega}}_P$ is the particles' normalized rotation axis. The dot product with $\bar{\boldsymbol{\omega}}_P$ ensures that only the vortex particles' direction is considered, and the kernel is normalized by the sum of desired vorticity. We achieve an exact regulation of the vorticity sum under the kernel in one timestep by this process.

Vortex particle seeding As explained in 3.2, particles will be seeded in regions of high normalized Reynolds stress. Based on the probability $p_T(\mathrm{x})$ from Eq. (8), a particle is created at position $\mathbf{x}$. All confined vorticity $\boldsymbol{\omega}_{ABL}$ within the particle's radius is removed from the artificial boundary layer, and the particle's strength and direction $\boldsymbol{\omega}_P$ are set such that the vorticity integrated over the kernel

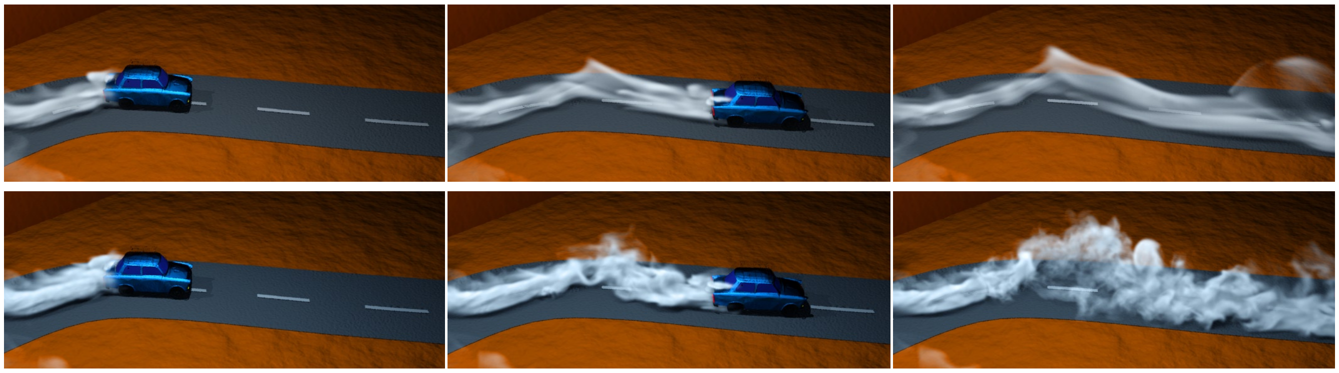

Figure 6: Example of a moving object inducing turbulence in its wake. The top row of images shows a simulation without vortex particles, the bottom row was simulated using our method.

equals the removed vorticity sum. We choose the particle radius to be as large as possible without touching an object. We allow for radii up to a size r_{max} which is fully resolved by the main simulation (we have used a value of $r_{max} = 6\Delta x$ below).

The constant c_P in Eq. (8) controls the granularity of the seeding process. If set to a high value, confined vorticity is turned into free turbulence relatively quickly. This results in a large number of weaker particles near the object, which then merge to large vortices. On the other hand, if c_P is set to a high value, the artificial boundary layer plume can grow, and fewer, stronger particles form. With an appropriately chosen c_P, numerical cost can be kept low while avoiding popping artifacts that may occur if overly large particles are seeded. We use a c_p of $\approx 1-4$ in our simulations.

5 Implementation

5.1 Precomputation

For precomputing the artificial boundary layer, it is essential to resolve the mean flow around an object. This can be done using time-averaging over a long period of time with a standard solver, or using a RANS solver. As we are only interested in the velocities around the boundary layer, we have used a standard solver with an artificially increased viscosity in the form of a diffusion step for the velocities. Due to the increased viscous effect, it stabilizes quickly and an average over fewer frames can be used. We have found that using the more complex RANS or longtime-averaging does not pay off visually compared to this more efficient solution. After obtaining the averaged flow field, the boundary layer is calculated according to the pseudo-code (Fig. 5) and stored as a point set.

Moving objects To precompute the flow for scenes with moving objects, the boundary layer around each moving objects is precalculated. If the object can move and rotate freely, or its movement is not known a priori, our algorithm allows us to precompute the whole range of movement directions to later on generate arbitrary simulations of the object in a flow. For this, we split the movement into a translational and a rotational component and precompute artificial boundary layers for each.

To perform the precomputation, the object is placed in the center of a simulation grid. The domain box is chosen large enough not to disturb the flow around the object. For the translational component, we leave the object fixed and use different inflow velocities, defined as boundary conditions on the domain box. As $\boldsymbol{\omega}_{ABL}$ is linear in the velocity magnitude, we only need to sample the velocity direction. In our simulations, we use 10×20 samples in spherical coordinates. For the rotational component, the object is placed in a standing fluid, and we rotate the object with normalized speed around a chosen axis. Again, we use 10×20 samples in spherical coordinates to sample the rotation axis direction.

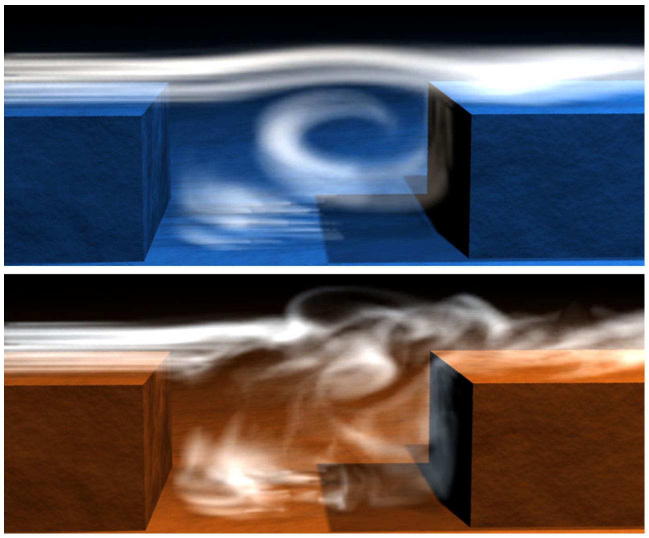

Figure 7: The top picture show a basic simulation of a fluid flowing left to right over a cavity. This flow produces a big vortex in the cavity, but is unable to capture any generation of turbulence from the walls. With our method (at the bottom) we are able to identify the confined vorticity shedding off the two edges of the cavity, and introduce corresponding vortex particles to represent the turbulent structures forming in the flow.

The simulations stabilize quickly due to the increased viscosity. We have used 50 steps for the examples shown in our video. In the precomputations for the rotational component, this is equivalent to one full rotation. After stabilizing, we average the velocities over another 50 frames. We note that precomputations for each direction can be trivially done in parallel.

Applying the precomputed set At simulation time, we determine the objects linear velocity relative to the scene, and its rotation axis. We then look up the nearest values in the precomputed database. A bilinear spherical interpolation is performed for both the linear velocity direction as well as the rotation axis. The results of the interpolation are scaled by respective magnitude and added. We have performed error measurements for the linear interpolation of the boundary layer values. The corresponding graph can be seen in Fig. 12. Our choice of 20 directional samples in the azimuth means we have an interpolation error of 1.6%. As the decomposition into rotational and linear component is only an approximation, we have also measured its error for the car model (Fig. 6). In this

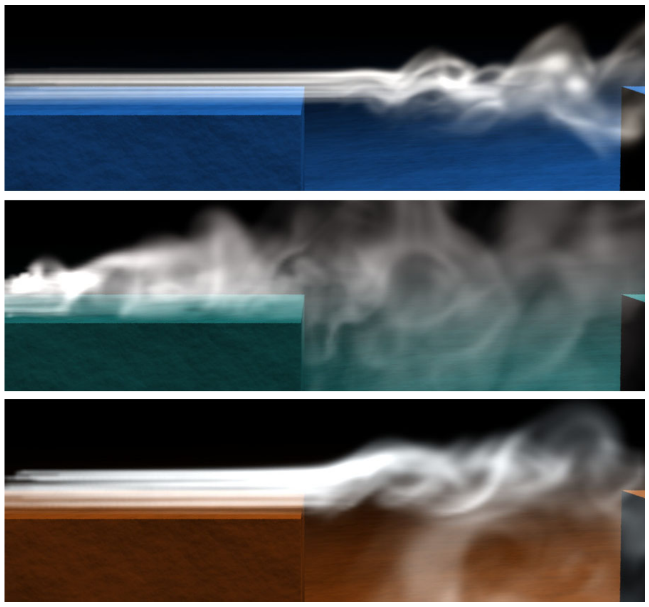
Figure 8: Comparison between a reference simulation (top picture), a simulation with randomly seeded vortex particles along the walls (middle), and our method (bottom). The random vortex particles destroy the overall flow structure, although the number and strength of the vortex particles are similar to those used in our method. Our approach correctly identifies the turbulence being shed off the step, similar to the reference simulation at the top, and introduces particles with a correct orientation locally into the flow.

```
 1: // Initialize boundary layer
 2: for each voxel $\mathbf{x}$ on an obstacle boundary do
 3:     Find corresponding $(\mathbf{x}_{pre}, \boldsymbol{\omega}_{pre})$ in precomputed set
 4:     // Initialize wall-attached ABL
 5:     $\boldsymbol{\omega}_{ABL}(\mathbf{x}) \leftarrow \max(\boldsymbol{\omega}_{ABL}(\mathbf{x}), \boldsymbol{\omega}_{pre})$
 6: end for
 7:
 8: // Simulate boundary layer seperation
 9: Advect $\boldsymbol{\omega}_{ABL}$ with the main flow
10:
11: // Seed vortex particles
12: for each voxel $\mathbf{x}$ with $\boldsymbol{\omega}_{ABL}(\mathbf{x}) \neq 0$ do
13:     $p_T \leftarrow 2\, c_P\, \Delta t\, (l_m\, |\boldsymbol{\omega}_{ABL}(\mathbf{x})| / |\mathbf{U}_0|)^2$
14:     if random() $< p_T$ then
15:         $(Y) \leftarrow$ voxels within particle radius of $\mathbf{x}$
16:         $\boldsymbol{\omega}_S = \sum_{(Y)} \boldsymbol{\omega}_{ABL}$    // sum within particle radius
17:         $\boldsymbol{\omega}_{ABL}(Y) \leftarrow 0$    // remove vorticity from ABL
18:         Seed particle at $\mathbf{x}$ with total vorticity $\boldsymbol{\omega}_S$
19:     end if
20: end for
21:
22: // Vortex particle dynamics
23: Advect vortex particles
24: Merge, split, dissipate vortex particles (§ 4.2)
25: Synthesize turbulence (§ 4.3)
26:
27: // Standard fluid simulation steps
28: Velocity self-advection, pressure projection etc.
```

Figure 9: Pseudo-code for a main simulation loop including our turbulence model.

case the error is 8% on average, and thus small enough not to cause visual artifacts.

It is not necessary to fully resolve the boundary layer during the precomputations, as our model described in § 3.1 takes care of this. Instead, one should make sure the resolution of the precomputation is sufficiently fine to resolve all important geometric features of the object. This is eased by the precomputation focusing only on that object, even if it will only occupy a tiny fraction of the final simulation domain. In addition, since the precomputation can be used on many simulations, a high resolution precomputation grid can quickly pay off. In § 6 we demonstrate the effectiveness of the precomputation even when an object is extremely thin.

5.2 Simulation loop

For the actual simulation, a standard fluid solver and a vortex particle system are coupled. In each simulation step, the artificial boundary layer is updated, new vortex particles are created and vortex particle dynamics are applied. Afterwards, the turbulence forces are added to the flow field, and finally, the remaining steps of the standard fluid simulation are performed. This is repeated each timestep. Pseudo-code for this extended simulation loop can be found in Fig. 9

Note that we can also independently choose a higher grid resolution for the evaluation of the vortex particles. This allows us to more accurately evaluate the particle kernels, which is especially useful for small scale vortex details. This high resolution velocity field is down-sampled for the main simulation steps (line 28 in the pseudo-code), and up-sampled for our algorithm before starting with line 1. Performing the algorithm (line 1–25) with a higher resolution enables us to simulate detailed features, e.g., when advecting smoke densities or a free surface level set, while the costly pressure projection operates on a small grid resolution. Typically, we have used a two times higher resolution for the examples below.

6 Results and Discussion

In the following section we discuss comparisons of our method to previous work and a reference simulation. In addition, we demonstrate several complex examples of turbulence being generated around moving objects or due to effects such as wind or a flowing river.

Comparisons In Fig. 7 the effect of our wall-induced turbulence can be seen for the flow over a cavity with a grid resolution of $120 \times 60 \times 40$. The top image shows a standard, unmodified simulation, while the lower image uses our algorithm to introduce wall-induced turbulence. Both simulations use the same grid resolution, but the unmodified simulation is unable to capture any turbulence being generated from the shearing near the walls. Our simulation exhibits complex vortices due to the vorticity generated at the wall boundaries. To compare our method to approaches for synthetic detail generation, we have simulated the same cavity setup including wavelet turbulence, using the implementation available on the paper's website [Kim et al. 2008b]. Wavelet turbulence successfully adds small detail to the overall flow, but has difficulties introducing larger vortices to the strong horizontal motion. In contrast, our method introduces persistent larger vortices, while the resulting smoke filaments are successfully broken up by the wavelet turbulence. This shows that our method is suitable for bridging the gap between small synthetic vortices and the vortices resolved by a standard simulation.

We use the setup shown in Fig. 8 to compare our method with normal vortex particles [Selle et al. 2005]. The simulations now focus on the left edge of the cavity. The top image shows a reference simulation, using a four times increased grid resolution. Note that the flow along the wall to the left is completely straight, while turbulent structures form to the right of the backward facing step. This behavior has been confirmed in various experiments and sim-

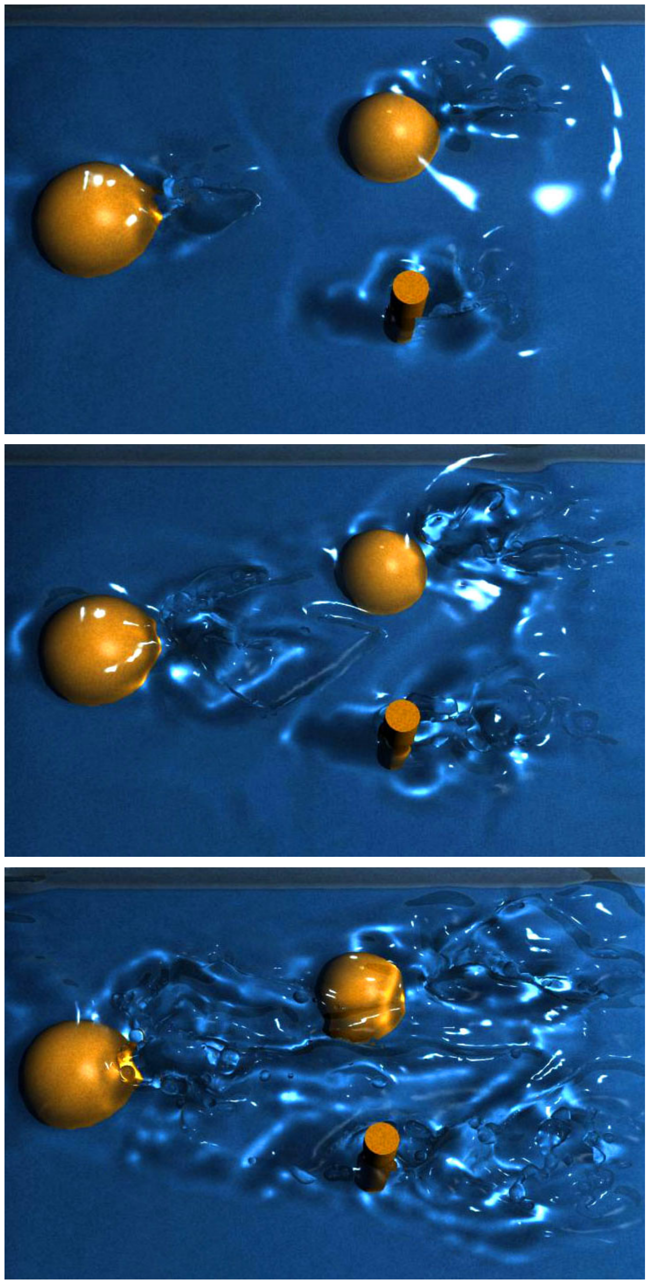
Figure 10: Our algorithm naturally extends to simulations of liquids. Here, we apply our algorithm to a river flow around three obstacles, resulting in turbulent wakes behind them.

ulations (e.g. [Le et al. 1997]). The middle image shows the flow after randomly introducing vortex particles along the walls. Naturally, the vortex particles do not take the overall flow into account, and strongly distort the structure of the flow. Our method, shown in the bottom picture, is able to recover the vortices being shed off the step, without distorting the flow along the wall to the left. Although we are not able to fully recover the flow of the reference simulation due to the different numerical viscosities, our method is able to qualitatively capture the wall-induced turbulence at a much lower computational cost. On average, the time per frame for the reference simulation was 218 times higher than for the simulation with our algorithm.

The limitation that motivated vortex particles was that vorticity confinement uniformly amplified vorticity magnitude. Our method, like vortex particles, overcomes this by allowing local modeling of vorticity, including effects like tilting and stretching. However, by modeling the boundary layer and considering the directions of particles vortices with Eq. (1), we are able to keep vortex particles from disturbing the bulk flow. This allows us to use vortex particles of larger magnitude than the randomly seeded vortex particles.

Complex Examples Next we consider examples with more complex geometry. Fig. 6 shows the simulation of a moving car that is emitting smoke. It can be observed how our model reproduces the dependence of turbulence strength from the cars velocity. As can be seen in the top row of Fig. 6, a normal simulation of the same resolution would not resolve any shed vortices at the car's surface. Second, Fig. 1 shows a thin whisk geometry stirring smoke. The boundary layer precomputation was done with a high resolution grid that resolved the whisk's wires, while its subsequent use in a smoke simulation was done on a much coarser grid. The standard grid did not resolve the wires, and only approximate velocity boundary conditions were set, resulting in the fluid slightly following the whisk's motion. Still, our algorithm was able to accurately generate vortices that are produced by its motion.

In Fig. 11 we show how our method works in conjunction with static flow fields. In this case we precompute a snapshot image of static flow around the object, and use it to advect the boundary layer, the vortex particles and the smoke densities. This simple form of simulation works without an expensive pressure correction step. Despite the simple underlying setup, we are able to produce complex structures forming in the wake behind the obstacle from the interactions of the vortex particles amongst themselves. Lastly, we demonstrate that our method can be easily extended to free surfaces in Fig. 10. Here three obstacles in the liquid produce turbulent wakes behind them. For this simulation, a particle level set [Enright et al. 2002] was used to represent the liquid's surface. Similar to particles near obstacle walls, we reduce a particle's kernel size once it extends past the liquid phase to avoid non-divergence free velocity fields.

Detailed grid sizes and timings for the examples above can be found in Table 1. The performance was measured on an Intel Core i7 CPU with 3.0 GHz. The majority of the time used for our approach (denoted by *ABL time* in Table 1) is taken up by the advection of the artificial boundary layer. For the liquid example of Fig. 10, the performance is strongly dominated by the particle level set. Overall, we achieve computing times ranging from 10 to 20 seconds per frame on average. An exception is the example with a static flow field, which requires only 1.3 seconds per frame.

Limitations A limitation of our method is that our precomputation currently assumes a rigid object, making it difficult to apply it to deforming objects such as cloth. To resolve this, a RANS solver could be coupled to a normal fluid solver to determine the current shear stresses at the object surface. Alternatively, it may be possible to precompute suitable boundary layer data for deforming objects by making use of data compression schemes. Also, our approach for the precomputation assumes the flow around the object can be described by the translational and rotational velocity components. If the flow around the object varies strongly, e.g., due to strong external forces or due to multiple objects in close vicinity, the resulting confined vorticity can differ from the desired values .

Extending our approach to handle this more accurately is interesting future work. In addition, a trade-off of our method is the use of vorticity reconstruction at a higher resolution. While this allows us to go beyond the coarse simulation Nyquist limit and get higher resolution detail (a limitation of the original vortex particle method), it means the domain and object boundaries as well as the free-surface boundary conditions are not as well modeled by the reconstructed high resolution velocity field.

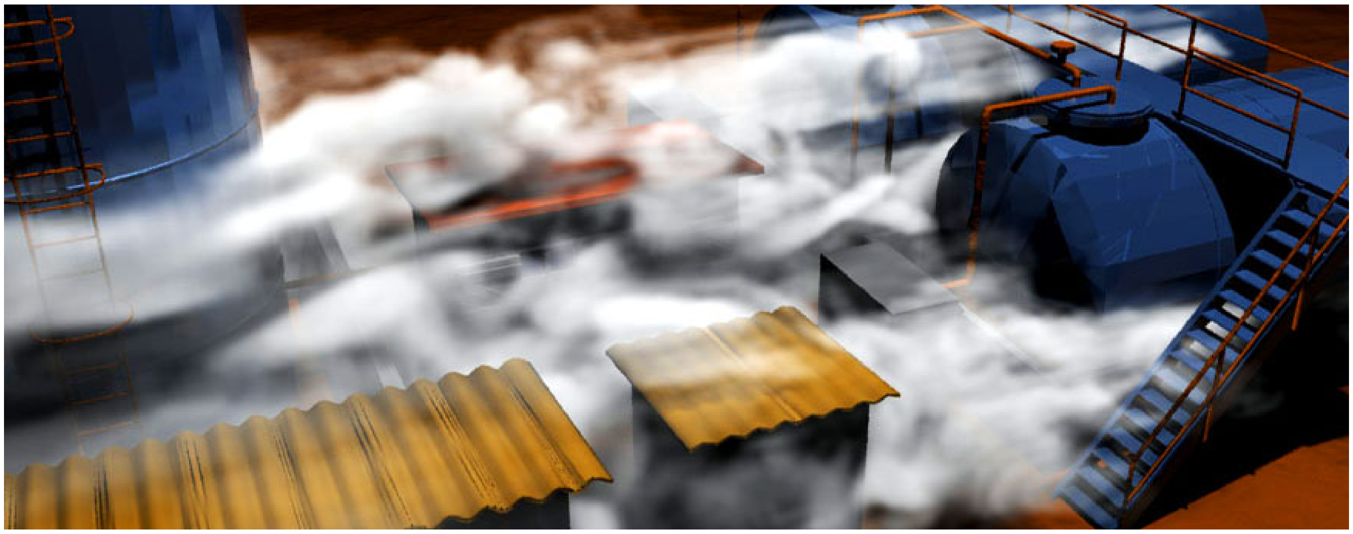

Figure 11: In this example a static flow field is used to generate complex turbulence around a object with our method.

Setup	Fig. 1	Fig. 6	Fig. 10	Fig. 11
Grid res.	160·70·160	150·40·200	100·25·60	250·80·150
ABL upscaling	2	2	2	1
Frame time [s]	19.5	13.4	10.0	1.3
ABL time [s]	5.5	3.7	0.05	0.7
# particles	$\sim$900	$\sim$700	$\sim$600	$\sim$1000
Vortex gain β	6.2	0.4	3.2	4.0
Precomp. res.	70x150x70	70x70x120	100x25x60	250x80x150
Precomp. [s]	220	112	59	227

Table 1: Detailed statistics for our simulation runs. *ABL upscaling* refers to the up-sampled grid on which vortex particle evaluation and smoke/levelset advection is performed. The precomputation time is given per database parameter.

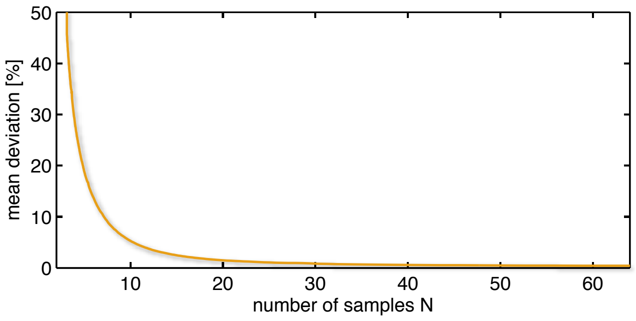

Figure 12: Mean relative error of the artificial boundary layer values for the car model. A reference simulation is compared to a spherical interpolation with N samples for the azimuth.

7 Conclusions

We have presented an algorithm for simulating wall-induced turbulence in fluid simulations. By modeling turbulence generation with the mean flow field, we are able to create a precomputed artificial boundary layer that captures the characteristics of turbulence generation around objects. Many simulations can then include this object and associated precomputed data. In particular, we determine transitioning regions and introduce appropriate vortex particles to represent turbulence. The particles are then evolved according to the vortex equations of flow to respect energy conservation and cascading. This yields the ability to efficiently compute physically plausible simulations of turbulence around rigid objects in a variety of settings.

In the future, we believe our approach will be useful in real-time settings, where a full fluid simulation is often too costly. In particular, promise is shown by Fig. 11, where complex vortex motion is produced without an expensive full simulation. We are also interested in continuing the trend of *multi-scale physics* by more closely coupling synthetic flow detail methods with our approach. This would yield a unified three-tier approach to modeling all scales of turbulence.

Acknowledgements The authors would like to thank the SIGGRAPH Asia reviewers for their comment and suggestions. Tobias Pfaff is supported by the ETH grant *TH-23 07-3*.

References

Baldwin, B. S., and Lomax, H. 1978. Thin Layer Approximation and Algebraic Model for Seperated Turbulent Flows. *American Institute of Aeronautics and Astronautics Journal*.

Batty, C., Bertails, F., and Bridson, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics 26*, 3, Article 100.

Bridson, R., Houriham, J., and Nordenstam, M. 2007. Curl-noise for procedural fluid flow. *ACM SIGGRAPH papers 26*, 3, Article 46.

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.

Driest, E. R. V. 1956. On turbulent flow near a wall. *J. Aeronaut. Sci. 23*, 11, 1007–1011.

Enright, D., Marschner, S., and Fedkiw, R. 2002. Animation and rendering of complex water surfaces. In *Proceedings of ACM SIGGRAPH*, pp. 736–744.

Fedkiw, R., Stam, J., and Jensen, H. W. 2001. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH*, 15–22.

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

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3, 871–878.

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.

JIMÉNEZ, J., AND ORLAND, P. 1993. The rollup of a vortex layer near a wall. *Journal of Fluid Mechanics*.

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.

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

KOLMOGOROV, A. 1941. The local structure of turbulence in incompressible viscous fluid for very large reynolds number. *Dokl. Akad. Nauk SSSR 30*.

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

LE, H., MOIN, P., AND KIM, J. 1997. Direct numerical simulation of turbulent flow over a backward-facing step. *J. Fluid Mech. 330*, 01, 349–374.

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).

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.

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*.

ROBINSON-MOSHER, A., SHINAR, T., GRETARSSON, J., SU, J., AND FEDKIW, R. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM SIGGRAPH papers 27*, 3 (Aug.), Article 46.

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*.

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.