Lagrangian Vortex Sheets for Animating Fluids

Tobias Pfaff
ETH Zurich

Nils Thuerey
ScanlineVFX

Markus Gross
ETH Zurich

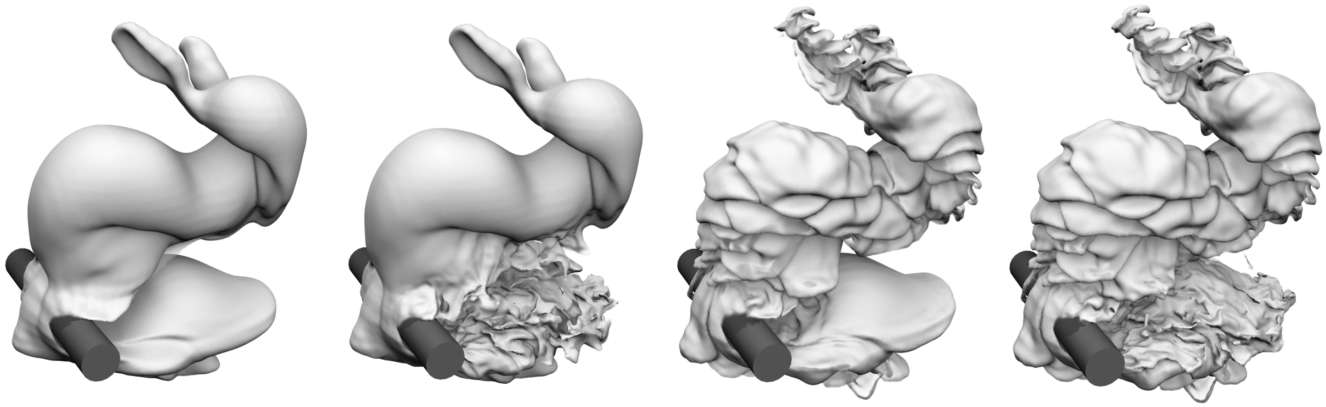

Figure 1: *A dense cloud subject to buoyancy forces and interaction with a moving obstacle is simulated. We use a Eulerian solver to compute a base flow, as shown on the left. Small-scale detail is synthesized directly on the interface of the cloud. An adapted turbulence model provides details from obstacle interaction (middle left), while small-scale buoyancy effects are calculated using vortex sheet dynamics (in the middle right). The picture on the right shows the combined model.*

Abstract

Buoyant turbulent smoke plumes with a sharp smoke-air interface, such as volcanic plumes, are notoriously hard to simulate. The surface clearly shows small-scale turbulent structures which are costly to resolve. In addition, the turbulence onset is directly visible at the interface, and is not captured by commonly used turbulence models. We present a novel approach that employs a triangle mesh as a high-resolution surface representation combined with a coarse Eulerian solver. On the mesh, we solve the interfacial vortex sheet equations, which allows us to accurately simulate buoyancy induced turbulence. For complex boundary conditions we propose an orthogonal turbulence model that handles vortices caused by obstacle interaction. In addition, we demonstrate a re-sampling scheme to remove surfaces that are hidden inside the bulk volume. In this way we are able to achieve highly detailed simulations of turbulent plumes efficiently.

CR Categories: Computer Graphics [I.3.7]: Animation—;

Keywords: Turbulence, Fluid Simulation

Links: DL PDF WEB

ACM Reference Format
Pfaff, T., Thuerey, N., Gross, M. 2012. Lagrangian Vortex Sheets for Animating Fluids.
ACM Trans. Graph. 31 4, Article 112 (July 2012), 8 pages. DOI = 10.1145/2185520.2185608
http://doi.acm.org/10.1145/2185520.2185608.

1 Introduction

When we look at fluid simulations in movies, arguably the visually most interesting scenes are those in which a lot of turbulent detail is visible. Smoke plumes from volcanoes, explosions or collapsing buildings are examples of highly turbulent flows, and the structure of the developing turbulent eddies is clearly visible at the sharp interface of the thick smoke and the air. At the same time, the thick clouds typically hide everything that is happening further inside the volume. Unfortunately, such scenes are numerically expensive to simulate, and quickly spend large amounts of computation on detail inside the cloud that will never be visible.

One way of dealing with the complex details of turbulent fluid simulation is turbulence modeling, which has increasingly been studied in computer graphics over the recent years. These methods are able to model and synthesize detail smaller than the simulation resolution, leading to faster run-times. However, even with a turbulence model the synthesized detail has to be represented in the simulation, and using a volumetric representation resolving the small-scale details requires immense storage capacity.

In our model, we therefore chose to explicitly discretize and track only the smoke-air interface. This greatly reduces the amount of information we need to store. In addition, this representation is a very suitable basis for turbulence methods. Instead of unnecessarily calculating detail that is hidden inside the smoke volume, we restrict synthesizing turbulence purely to the visible smoke interface.

The phenomena mentioned above exhibit another interesting effect: turbulence production in such flows mainly stems from buoyancy, which induces a vortex sheet at the smoke-air interface. This sheet reinforces small-scale surface instabilities, which then develop into turbulence. This means that the transition region where the turbulence is created is clearly visible, and this turbulent onset strongly influences the visible shape of the interface. However, the simulation resolution is typically too limited to directly capture these small-scale buoyancy effects. Furthermore, most turbulence models assume fully-developed homogeneous turbulence, which means they are valid inside the bulk smoke volume, but not at the inter-

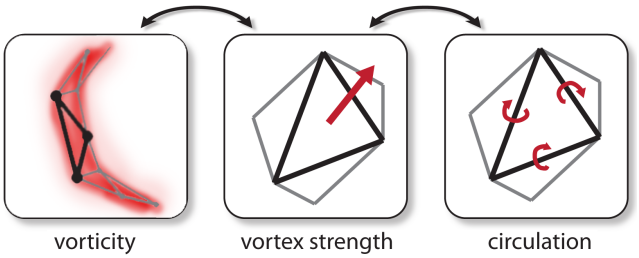

Figure 2: *The continuous vorticity field around a surface can be represented in terms of a vortex sheet strength or circulation. Both are stored per surface triangle, and are equivalent representations. Vortex strength is a vector value, while circulation consists of three scalar rotation values around the edges of the triangle.*

face. Here, the turbulence generation process is highly anisotropic and model-dependent in nature. This means it is not well described using the statistical approaches that are the basis for most turbulence methods.

Our method addresses this problem by directly tracking the vortex sheet at the smoke-air interface. This allows us to compute buoyancy effects at scales independent of an underlying grid, and accurately model the turbulence generation process due to buoyancy. Traditionally, correct handling of obstacles is very difficult for vorticity based methods. Our model handles basic interaction with static or moving obstacles using a Eulerian solver, and we capture the turbulence they induce with a model specifically tailored to our needs. We will ensure that the turbulence model for obstacles is orthogonal to our buoyancy approach, which makes it possible to use both in combination or separately as needed.

In summary, we propose an algorithm with the following contributions:

- A *local evaluation scheme* for vortex sheets which allows us to efficiently capture detailed buoyant and obstacle based turbulence effects.
- A *turbulence model for obstacles* that is able to estimate wall-induced turbulence and is orthogonal to buoyancy based turbulence.
- A *mesh resampling* technique for efficiently pruning invisible detail to reduce mesh complexity.

We use an adaptive triangle mesh to simulate non-diffusive smoke surfaces, and couple it to an Eulerian solver which captures the large-scale motion of the flow. We will demonstrate that this representation is very suitable for vorticity based methods and that it produces highly detailed visuals efficiently.

2 Related Work

For large parts, fluid simulation in computer graphics falls into the categories of Eulerian solvers with semi-Lagrangian advection as introduced by Stam[1999], and Lagrangian approaches, such as Smoothed-Particle Hydrodynamics methods, e.g. [Müller et al. 2005]. Our representation of vortex sheets is independent of the solver type, but we will focus on grid based solvers due to their wide-spread use.

An intrinsic problem associated with fluid simulation is the representation of detail without having to resort to costly high-resolution simulations. In particular, this problem is due to damping of the velocity field and all other data represented on the grids. A popular approach to alleviate this problem is to use higher order advection schemes such as MacCormack advection [Selle et al. 2008], or the commonly used FLIP [Zhu and Bridson 2005] model. In addition, Mullen et al. [2009] introduced an integration scheme that preserves energy. While these methods help to reduce the numerical dampening, the detail that can be represented is still inherently limited by the underlying grid resolution.

One possible solution is to adaptively refine the simulation grid, as was done, e.g. in [Losasso et al. 2004], or more recently in [Chentanez and Mueller 2011]. These methods can pay off when the detail is confined to small parts of the computational domain. For liquids, particle level sets [Enright et al. 2002] increase the resolution of a level set using Lagrangian markers, while Bargteil et al. [2006] and Wojtan et al.[2010] use a triangle mesh to represent liquid-air interfaces. Particles are another popular choice, but large numbers are usually necessary to represent dense surfaces without noise. While the Lagrangian markers in these methods allow for the detailed representation on sub-grid scales, the dynamics are still limited by the grid resolution of velocity field. Similar in spirit to our approach, Brochu et al. [2009] use a triangle mesh to represent detailed smoke structures, however, without leveraging this representation to compute additional dynamics.

Turbulence models obtain small-scale dynamics by modeling, instead of simulating it. Early methods synthesized a divergence-free turbulence field using the Kolmogorov spectrum, e.g, in [Stam and Fiume 1993], and [Rasmussen et al. 2003], while recent methods measure the turbulent energy spectrum [Kim et al. 2008] or model turbulent energy transport to obtain spatially correct turbulence distributions [Narain et al. 2008], [Schechter and Bridson 2008], [Pfaff et al. 2010]. These approaches work well to model fully developed turbulence in the bulk flow. However, due to their nature they cannot capture turbulence transition and the onset of turbulence, which are important effects that are clearly visible at the interface of a dense cloud.

Vortex methods, on the other hand, use the vorticity formulation of the Navier-Stokes equation to compute fluid dynamics. This has the advantage that detailed, turbulent motion is well described by a vorticity formulation. In vortex methods for graphics, this vorticity is usually stored with sparse Lagrangian elements. Vortex particles [Selle et al. 2005], [Pfaff et al. 2009] can be used to augment a grid-based fluid simulation with turbulent detail. Angelidis et al. [2006] use vortex filaments to control a fluid simulation, while Weissmann and Pinkall [2010] propose a simulation driven entirely by sparse filaments. With these approaches, re-meshing sparse particles and filaments such that the turbulence characteristics are preserved is hard.

In contrast to our approach, none of the methods deals with buoyancy-driven vorticity, or *baroclinic* vorticity, as it is commonly called in literature. Kim et al. [Kim et al. 2009] use a vortex sheet formulation to reinforce the breakup of liquid sheets. However, they discretize vorticity on the grid, and synthesize motion using Eulerian vorticity confinement. In summary, few existing methods in graphics are suitable for simulating the turbulence transition observed in turbulent, buoyant smoke, as they usually focus on fully-developed turbulence and do not model sub-grid baroclinity.

Vortex methods for simulating turbulent flows are a common research topic in the CFD community. While a large part of these works focus on the more traditional vortex particle representation, as introduced by Rosenhead [1931], and filament methods [Leonard 1980], vortex sheet methods have become increasingly popular in recent years. Vortex sheet methods operating on connected triangles or quad representations similar to our approach include [Brady et al. 1998], [Stock et al. 2008] and [Lozano et al. 1998], to mention a few examples from the growing body of work. Similarly, baroclinic generation has been studied for engineering applications in [Meng 1978] and [Tryggvason and Aref 1983].

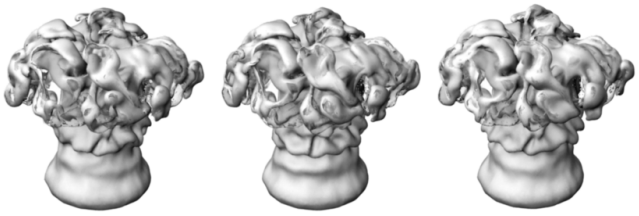

Figure 3: *A buoyant plume is simulated without evaluation cutoff (left), with a cutoff of 10 cells (middle) and 5 cells (right, our default setting). While details are different due to accumulation of small differences over time, the visual quality is comparable.*

Our work is inspired by mesh-based vortex sheet methods from CFD, but differs in several important aspects. While methods such as [Stock et al. 2008] are highly suitable to accurately capturing small-scale phenomena, they focus on the ideal case of pure buoyancy driven flows and cannot handle scenes with static walls or moving obstacles, such as Fig. 9 or Fig. 1. In addition, we will later on show that even for pure buoyancy driven flows our approach yields a significant speedup over the a classical vortex sheet method thanks to its local evaluation.

In the following section, the established theory on vortex sheets will be introduced in more detail, while we will focus on our extended model in the subsequent sections.

3 Vortex Sheet Methods

Fluid solvers in graphics typically use the velocity formulation of the Navier-Stokes (NS) equations to obtain the fluid motion. However, when considering turbulence the vorticity formulation of the NS equations is advantageous. The vorticity $\boldsymbol{\omega}$ corresponding to the velocity field $\mathbf{u}$ is given by $\boldsymbol{\omega} = \nabla \times \mathbf{u}$. The inviscid NS equations without external forces therefore transforms to

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{d}t} = \boldsymbol{\omega} \cdot \nabla \mathbf{u} + \frac{1}{\rho}\nabla\rho \times \left(\mathbf{g} + \frac{1}{\rho}\nabla p\right) \tag{1}$$

with density ρ, pressure p and gravity $\mathbf{g}$. The total derivative on the left-hand side includes vorticity advection, while the right-hand side consists of the vortex stretching term and the often neglected baroclinity term. This formulation does not require solving a Poisson problem to make the velocities divergence free. Instead it requires additional work to reconstruct the velocity $\mathbf{u}$ from the vorticity.

For accurately discretizing the vorticity form of the NS equations on our surface meshes we make use of three different representations. Apart from the aforementioned vorticity $\boldsymbol{\omega}$ we will introduce a vorticity confined to surfaces: the *vortex sheet* strength, and the *circulation*, which is vorticity confined to lines. The differences are illustrated in Fig. 2. During discretization, we will use each representation where it is most suitable.

Vortex sheet strength Apart from external sources, the only vorticity source in Eq. (1) is the baroclinity. In a system of two fluids with different densities ρ_1 and ρ_2 we will therefore observe vorticity forming due to the density gradient at the interface of these two fluids. To track this vorticity, we can use an explicit representation of their interface. On this vortex sheet, vorticity associated with the density gradient accumulates. A vortex sheet is defined by the vortex strength $\boldsymbol{\gamma}$ which relates to vorticity as $\boldsymbol{\omega} = \boldsymbol{\gamma}\,\delta(n)$ with Dirac's delta function δ around the surface n. We can now formulate Eq. (1) in terms of this vortex strength [Wu 1995]

$$\frac{\mathrm{D}\boldsymbol{\gamma}}{\mathrm{d}t} = \boldsymbol{\gamma} \cdot \nabla \mathbf{u} - \boldsymbol{\gamma}(\mathbf{P} \cdot \nabla \cdot \mathbf{u}) - 2\beta\,\hat{\mathbf{n}} \times \mathbf{g} \quad . \tag{2}$$

The first term on the right-hand side is the familiar vortex stretching, while the second term describes changes in vortex strength due to elongation in the direction of $\boldsymbol{\gamma}$. Here, $\mathbf{P} = \mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}$ is the tangential projection operator with the surface normal $\hat{\mathbf{n}}$. The last term represents baroclinity with the Boussinesq approximation[Meng 1978] which is proportional to the Atwood ratio β. The Atwood ratio relates the densities of the two fluids to each other, and is defined as $\beta = (\rho_1 - \rho_2)/(\rho_1 + \rho_2)$. Note that β allows for an easy way to artistically control the strength of the formation of baroclinic turbulence. Although this value is constant in our scenes, it would be easy to track spatially varying values on the mesh. These values could, e.g., be painted on the initial surface by an animator. It should also be noted that the Boussinesq approximation in Eq. (2) assumes a small Atwood ratio, and is valid for e.g. hot/cold air, but not air/water interfaces. To summarize, vortex sheets are very suitable for describing fluid phenomena dominated by baroclinic generation, as the vorticity stays concentrated at the density interface.

Velocity integration To evolve the flow with Eq. (2), we need to obtain the velocity field $\mathbf{u}$. Here, we use the free-space solution to the rotation operator, the Biot-Savart law, to integrate the velocity field induced by the vortex sheet $\boldsymbol{\gamma}$:

$$\mathbf{u}(\mathbf{x}) = \frac{1}{4\pi}\int \boldsymbol{\gamma}(\mathbf{x}') \times \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|^3} d\mathbf{x}' \quad . \tag{3}$$

Discretization To solve the vorticity dynamics equations, it is necessary to have a discretization of the interface. For this we use a mesh consisting of triangles, where each triangle i has a corresponding vortex strength $\boldsymbol{\gamma}_i$. The evolution of $\boldsymbol{\gamma}_i$ over time is calculated by integration of Eq. (2) over time. However, for the evaluation of vortex stretching and elongation it is advantageous to switch to a formulation based on the circulation. The scalar-valued circulation Γ_a defines a rotation around the axis of edge a. According to Stock et al. [2008], the circulations around the 3 edges $\mathbf{e}_{1,2,3}$ of a triangle with area A uniquely relate to its vortex sheet strength as

$$\boldsymbol{\gamma} = \frac{1}{A}\sum_{i=1}^{3} \Gamma_i\,\mathbf{e}_i \quad . \tag{4}$$

On the other hand, to obtain circulation from vortex strength, the overdetermined linear system

$$\begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \end{pmatrix} = A \begin{pmatrix} \boldsymbol{\gamma} \\ 0 \end{pmatrix} \tag{5}$$

is solved. We now take advantage of the fact that the vortex stretching and elongation terms in Eq. (2) are implicitly handled in circulation notation. In other words, for a system without baroclinity, the evolution equation for circulation reduces to $\mathrm{D}\Gamma/\mathrm{d}t = 0$. We therefore solve the advection of the vortex sheet in terms of circulation, and switch back to the vorticity formulation for adding the baroclinity term, and the integration of velocities. As the translation is performed on a per-triangle basis, the three circulation values $\Gamma_{1\ldots3}$ are stored for each triangle in addition to the vortex strength. [1]

[1] We note that Γ retains the closedness condition during the simulation, i.e. $\sum_i \Gamma_i = 0$ for all triangles.

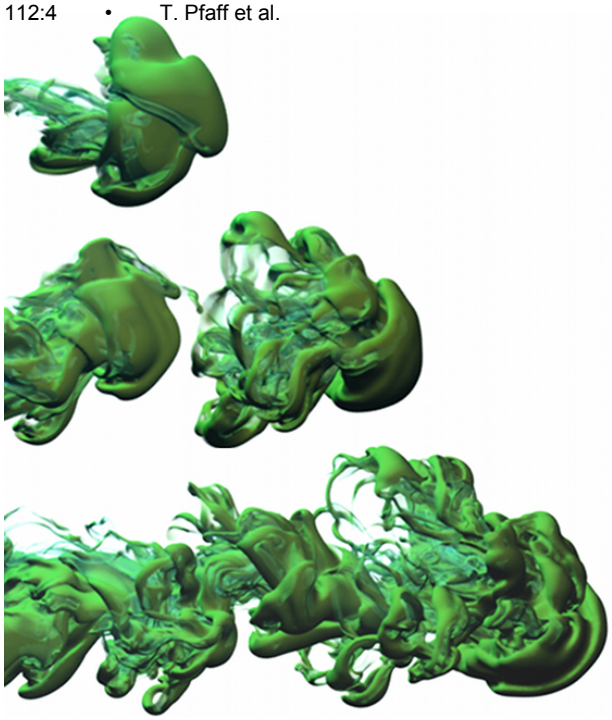

Figure 4: *We simulate the dynamics of a dense fluid in water with pulsed inflow conditions. The buoyancy leads to complex surfaces in the downstream region to the right.*

4 Local evaluation

Using the vortex strength $\boldsymbol{\gamma}$, we are able to obtain the buoyancy-induced velocity update by integrating Eq. (3). However, this velocity field only describes the effect of ideal buoyancy in free space, while most practical scenes have a nontrivial underlying flow due to obstacle interaction and boundary conditions. Also, the evaluation of Eq. (3) is very costly for large meshes. We therefore split our simulation into two parts: first, a Eulerian solver which computes a consistent flow field from obstacle interaction, inflows, and the large-scale effects of buoyancy. Second, a surface mesh which is used for front tracking of the smoke cloud and the simulation of detail due to small-scale buoyancy effects and obstacle turbulence.

Coupling Eulerian and Lagrangian Buoyancy For computation of the the large-scale flow, we use a standard grid-based solver [Stam 1999] with second order semi-Lagrangian advection as described in Selle et al. [2008]. Our vortex sheet approach enables us to use low grid resolutions, as details will be computed directly on the Lagrangian mesh. In the grid-based solver, a density field is tracked which is then used to compute coarse-scale buoyancy forces on the velocity field.

Evaluation of the small-scale buoyancy effects is performed using the vorticity of the mesh. To avoid duplication of buoyancy forces between grid and mesh, we remove the large-scale component of the baroclinic vorticity from the mesh. We first apply a Gaussian smoothing kernel on the vortex sheet strength $\boldsymbol{\gamma}$. The kernel width σ is set to match the grid cell width Δx to obtain the smoothed, grid-scale vortex strength component $\bar{\boldsymbol{\gamma}}$. The difference $\boldsymbol{\gamma}' = \boldsymbol{\gamma} - \bar{\boldsymbol{\gamma}}$ now represents the details below grid scale, which are evaluated on the mesh.

By removing the mean only the high-frequency variations $\boldsymbol{\gamma}'$ remain, whose effect decays very quickly in the far field. This corresponds to the formation of small vortices, which act locally. We are therefore able to introduce a cutoff radius r_C to the evaluation. Only triangles within this radius have to be evaluated in the summation of Eq. (8). As we can rely on the grid solver to capture the large scale buoyant motion, the effects of this approximation are negligible. A comparison of a full evaluation versus two different cutoff radii can be seen in Fig. 3. As the cutoff approximation introduces small differences which accumulate over time, the resulting surfaces differ. However, the visual quality is comparable for all three simulations, while the processing time is five times faster using $r_C = 5\Delta x$. We use this value for all following simulations with our model. The position update for the mesh nodes is performed based on the Eulerian velocity field, and by applying a per-node velocity update for the small-scale structures, which is described next. The complete simulation loop for our combined solver is summarized in pseudo code in Fig. 5.

```
// Grid-based Fluid solver
Semi-Lagrangian density and velocity advection
Add grid-based buoyancy
Pressure projection

// Turbulence model
Compute production: P_wall = 2ν_T |∇ × U − ω_g|^2
Update ω_g based on Eq. (10) and advect
Update k, ε based on Eq. (11) and advect

// Mesh dynamics
Integrate baroclinity: γ_i ← γ_i − Δt 2β n̂ × g
Compute Gaussian filtered vortex strengths γ̄_i
Small-scale vortex strength: γ'_i ← γ_i − γ̄_i

Compute circulations Γ_i ⇐ γ_i , Eq. (5)
for each mesh vertex i do
    u_i ⇐ Integrate Eq. (8) for sources γ'_i within r_C
    Advect vertex with u_i and grid velocity field
    Advect vertex with synthesized curl noise u_T = sqrt(ηk) y
end for
Compute vortex strengths γ_i ⇐ Γ_i , Eq. (4)

Perform mesh surface smoothing
Perform edge collapses and triangle subdivision
```

Figure 5: *Pseudo-code for the simulation loop of our algorithm.*

Regularization To obtain the small-scale velocity update for the mesh, Eq. (3) is discretized, using the residual vorticity $\boldsymbol{\gamma}'$ as a source. As this equation is singular for points on the interface, we chose to regularize the equation analogous to the vortex blob regularization for vorticity particles [Chorin and Bernard 1973]

$$\mathbf{u}_{reg}(\mathbf{x}) = \frac{1}{4\pi}\int_S \boldsymbol{\gamma}'(\mathbf{x}') \times f_{reg}(\mathbf{x}-\mathbf{x}')d\mathbf{x}' \qquad (6)$$

$$f_{reg}(\mathbf{r}) = \frac{\mathbf{r}}{(|\mathbf{r}|^2+\alpha^2)^{\frac{3}{2}}} \quad . \qquad (7)$$

The regularization parameter α effectively controls the minimal size of the generated vortices. We therefore set α proportional to the mesh resolution, as will be explained in § 6. To discretize this equation, we use Gaussian quadrature. If $G_j(\mathbf{r})$ is the Gaussian quadrature of f_{reg} for triangle j, Eq. (6) becomes

$$\mathbf{u}_i = \frac{1}{4\pi}\sum_{j=1}^{m} a_j \boldsymbol{\gamma}'_j \times G_j(\mathbf{r}_i) \quad . \qquad (8)$$

For each mesh node i, we therefore evaluate a sum over all triangles $j = 1 \ldots m$ which lie within the cutoff radius r_C. In our examples, we use three-point quadrature, and refer the reader to [Cowper 1973] for details on how to compute the integration weights.

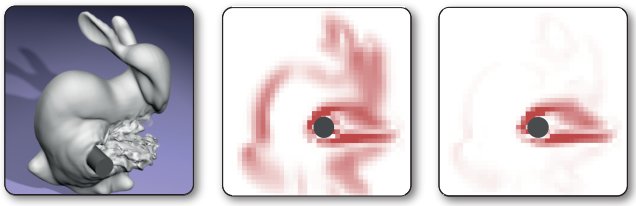

Figure 6: *To separate the sources of buoyancy and wall-based turbulence, buoyant vorticity is tracked over time. The total vorticity of a snapshot from Fig. 1 is shown in the middle picture, while the difference to the tracked buoyant vorticity is shown to the right. The gray circle marks the position of the cylinder. We observe that despite a small residual halo, our model tracks the area of obstacle influence behind the cylinder very well.*

5 Wall-based Turbulence Model

Our mesh representation also allows us to evaluate turbulence generated from interaction with obstacles directly on the interface. The turbulence model we propose in the following is orthogonal to the buoyancy model of the previous sections, and both models can be used independently or in combination. We first model the spatial and temporal distribution of turbulent kinetic energy k using an *energy transfer model*, and then synthesize turbulent detail on the surface using *frequency-matched curl noise*. Below, we will briefly outline the theory used, and explain our modifications. For a more in-depth account of turbulence modeling we refer the reader to the book of Pope [2000].

Modified Energy Model We compute the energy dynamics based on the commonly used k–ε model. It consists of two coupled PDEs that describe the evolution of a turbulent energy k and dissipation ε. Details of the model can be found in the Appendix A. The model can also be solved on the high-resolution surface mesh, but this did not yield a significant difference in our experiments. The reason for this is that the variables k and ε are averaged properties, and spatially vary smoothly due to turbulent diffusion. In the following, we assume the PDEs of Eq. (11) are solved on a grid for simplicity.

The primary interest here is to compute source terms for driving the model. The sources should capture the wall-induced turbulence, but exclude turbulence induced by buoyancy. If we were to directly use k for injecting turbulence we would include the effects of buoyancy twice - once from the k–ε model, and once from the vortex sheet model. In addition, a general turbulence model would not be able to capture the characteristic effects of buoyancy, such as the cloud billowing. We therefore need to guarantee orthogonality of the two methods, by excluding the effects of buoyancy from Eq. (11), such that each model can focus on the type of turbulence it is most suitable for. With a strain-based production term that is commonly used for the k–ε model, this would however imply separating the wall induced turbulence from the total one. This is, to the best of our knowledge, not possible for a strain based production. There is, however, an alternative production term $\mathcal{P}_R$ based on rotation. Compared to the strain based measure, it is less accurate for free-stream generation but still captures buoyancy and wall induced turbulence very well. Assuming we have a measure for the current buoyancy-induced turbulence, we can subtract it from $\mathcal{P}_R$ to single out the turbulence induced by obstacles. We have found that using the rotation-based production term from Spalart [1992] and a vorticity based integration of the buoyancy production allows us to do just this.

According to Spalart [1992], the production is given by $\mathcal{P}_R = 2\nu_T \sum_{i,j} \Omega_{ij}^2$, with the rotation tensor $\Omega_{ij} = \frac{1}{2}(\partial U_j/\partial x_i - \partial U_i/\partial x_j)$ of the large-scale velocity field $\mathbf{U}$. We now express its tensor norm in terms of vorticity as $\sum_{i,j} \Omega_{ij}^2 = |\boldsymbol{\omega}_f|^2$. Here $\boldsymbol{\omega}_f$ is simply the vorticity of the grid-based flow field given by $\boldsymbol{\omega}_f = \nabla \times \mathbf{U}$. With $\boldsymbol{\omega}_g$, which denotes the buoyancy induced vorticity strength that we will compute below, we obtain turbulence production for purely wall-generated turbulence using the difference of the two:

$$\mathcal{P}_{wall} = 2\nu_T |\nabla \times \mathbf{U} - \boldsymbol{\omega}_g|^2 \quad . \tag{9}$$

For stability, we ensure that $|\nabla \times \mathbf{U}| \geq |\boldsymbol{\omega}_g|$. An example from the simulation of Fig. 1 comparing the two vorticity measurements can be found in Fig. 6. Finally, we need to compute the accumulated vorticity induced by buoyancy $\boldsymbol{\omega}_g$. Applying the Boussinesq assumption and omitting external forces, we obtain an evolution equation for the buoyant vorticity $\boldsymbol{\omega}_g$ with

$$\frac{\mathrm{D}\boldsymbol{\omega}_g}{\mathrm{d}t} = \boldsymbol{\omega}_g \cdot \nabla \mathbf{u} + \frac{1}{\rho}(\nabla \rho \times \mathbf{g}) \quad . \tag{10}$$

We integrate this equation over time on the grid in combination with the k–ε model to obtain the wall based turbulence production $\mathcal{P}_{wall}$ as outlined in Fig. 5. Equipped with this production term we compute the spatial distribution of the turbulent kinetic energy k that we use to synthesize turbulent detail on the smoke surface.

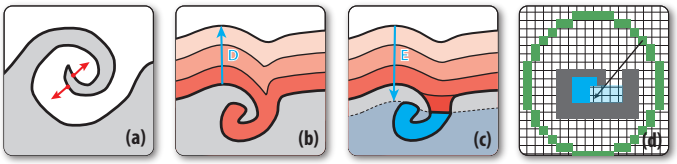

Figure 7: *To simplify mesh geometry, we collapse invisible thin sheets. We fist identify candidate nodes in very thin sheets (a). Next, we compute an eroded inside volume on grid in steps (b) and (c). Finally, we check whether these cells are visible with a raycast towards an enclosing sphere (d). All thin sheet nodes in the blue region of (d) are marked for edge collapses.*

Turbulence Synthesis In contrast to buoyancy induced turbulence, we can make use of Kolmogorov's famous five-thirds law for synthesizing the turbulence triggered by our k–ε model. In this regime energy is mainly scattered from large to small scales, so we can approximate the velocity of the turbulent details using a frequency-matched curl noise texture that is advected through the large-scale velocities, as in Kim et al.[2008]. Instead of evaluating the turbulence at each cell of a higher resolution grid, we can synthesize it more accurately on the mesh. Each mesh node carries a texture coordinate $\mathbf{q}$ for curl noise texture, and its turbulent kinetic energy k is interpolated from the grid. The additional velocity per node is then given by $\mathbf{u}_T = \sqrt{\eta k}\,\mathbf{y}(\mathbf{q})$, where $\mathbf{y}$ is the turbulence function from [Kim et al. 2008] and η is a scaling parameter that can be used to control bulk turbulence strength. We will demonstrate the interplay of the two turbulence models and their orthogonality in § 7.

6 Mesh Resampling

Due to advection and buoyancy, the mesh will undergo strong deformations. On the other hand, Gaussian smoothing and buoyancy integration rely on a relatively uniform mesh geometry. Therefore, we split and collapse triangle edges to keep all edge lengths l in the range $\Delta l < l < 2\Delta l$, where Δl is the desired minimal edge length. Vortical forces smaller this minimal length would only be visible as a slight noise on the surface. So we use the regularization parameter α in Eq. (6) to enforce a minimum vortex size larger than Δl. For our example scenes, we chose $\alpha = 2\Delta l$. Finally, we apply

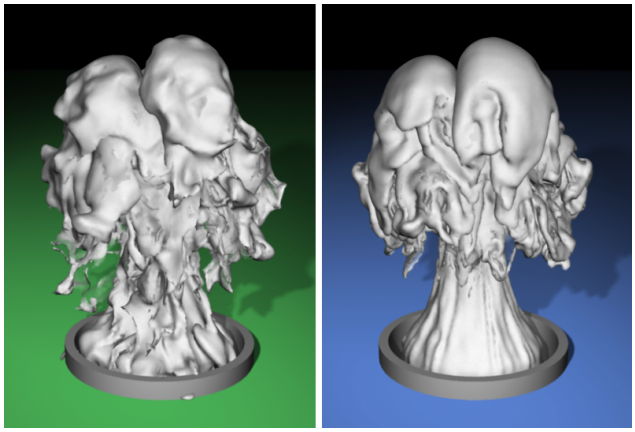

Figure 8: *We compare the simulation of a buoyant plume with isotropic turbulence modeling (left) to our method (right). While isotropic turbulence creates unrealistic surface distortions, the turbulence onset is calculated correctly using our approach.*

a small amount of explicit Laplacian smoothing to the mesh [Desbrun et al. 1999], to prevent the accumulation of small-scale noise on the surface.

The vortical motion on the mesh interface creates vortex roll-ups, which lead to the generation of spiral-shaped thin sheets. Since vorticity generation is linked to the surface normal, both sides accumulate almost equal amounts of vorticity, with opposing direction vectors. As the sheets become thinner, the vorticity effect on surrounding nodes therefore becomes smaller and effectively cancels out. Also, many of these thin structures are typically hidden inside the bulk volume of the cloud. Based on these two observations we propose the following algorithm to identify these sheets and remove the ones that are invisible from the outside. First, we mark nodes on thin sheets, check which of these are far inside volume, and finally perform a visibility test to determine nodes not visible from the outside. The process is visualized in Fig. 7.

As a first step, *thin sheet nodes* are identified by checking for a vertex with opposing normal ($\pm$ 20°) within close proximity, i.e. at a distance less than Δl opposing the vertex normal. This can be done efficiently using the grid as acceleration data structure. Next, we identify the volume inside the cloud on the grid. As a coarse representation of the outer hull, we first compute a level set for the mesh. Since triangle size is always well below the size of a grid cell, we can employ a simple and fast method [Kolluri 2005] to obtain the signed distance function. We then enlarge and shrink the level set to close small holes and cavities induced by the complex mesh geometry. The level set is enlarged by $D = 4$ cells to compute an outer interface. We rebuild the signed distance function at a distance $E = -(D + 2)$ from this interface, to obtain a faired volume slightly smaller than the original one. All cells inside this volume are marked as *inside cells*.

As cells in a cavity might still be visible from the outside, we finally compute visibility for the inside cells by performing a raycast towards target points on a sphere enclosing the surface mesh. The cost for these tests is less than 5% for our simulations, as there are typically few cells to be tested. All thin sheet nodes that are located in cells identified as not visible from the outside are marked to be collapsed during the next edge collapse step in line 25 of Fig. 5. For the example setup of Fig. 8, this method reduces the number of triangles by 32% at the end of the simulation, resulting in an overall speedup of 43%. We note that this reduction based on edge collapses could be improved, e.g., by using methods like [Wojtan et al. 2010], but we have found it to be efficient both in terms of stability as well as performance.

Figure 9: *An expanding, turbulent smoke front is simulated. Obstacle interaction is handled due to the coupling with a Eulerian solver.*

7 Results

In the following, we demonstrate the properties of our model based on several simulations setups. For most scenes we have used a shader that computes a transparency based on the length a ray spends inside the mesh volume. The only exception is Fig. 9, where we have rasterized the plume onto a grid data structure to make use of a volumetric shader that supports multi-scattering.

Turbulence onset To demonstrate the ability of our vortex sheet dynamics to correctly compute the turbulence onset, we simulated a buoyant smoke plume as shown in Fig. 8. The setup uses $64 \times 96 \times 64$ grid cells for the base solver, and a triangle edge length $\Delta l = 0.18\Delta x$. Without artificial disturbing forces, the base flow remains smooth and does not show any turbulent detail. To demonstrate the effect of standard turbulence methods, we synthesize turbulence using vortex particles. The vortex particles are emitted at the inflow and moved along the flow with the smoke plume. For the particles, we use a size and energy distribution based on the Kolmogorov spectrum. This is typically a good assumption for bulk volume flows, as isotropization drives the turbulence towards a Kolmogorov spectrum eventually. At the interface, however, the length scales are model-dependent and production is highly anisotropic. This leads to a lack of coherent features using isotropic turbulence methods. Using our method, we observe that the generated detail organically integrates with the large-scale flow.

Eulerian-Lagrangian coupling We demonstrate the generality of our model by simulating two setups with more complex boundary conditions. The first scene, depicted in Fig. 9, shows strongly billowing clouds moving through a channel of irregularly shaped obstacles. We simulate an expanding front of smoke with density slightly above air, with a base resolution of $40 \times 40 \times 128$. It can be seen that the flow easily follows the geometry of the scene due to the Eulerian simulation, while our vortex sheet model leads to the development of the typical billowing cloud surfaces. In the second scene, shown in Fig. 4, the interaction between water and a heavier liquid is simulated. We use a base solver with $96 \times 64 \times 64$ grid cells, and pulsed inflow conditions to simulate the injection of multiple drops of fluid. In this case, the temporally changing in-

Setup	Grid res.	#tris mio.	$\Delta l / \Delta x$	Mesh [s]	Grid [s]
Bunny Fig.1	$64 \times 64 \times 64$	0.9 / 2.6	0.2	9 / 33	0.6
Water Fig.4	$96 \times 64 \times 64$	0.8 / 3.2	0.15	12 / 40	1.3
Plume Fig.8	$64 \times 96 \times 64$	0.6 / 2.3	0.18	7 / 22	0.6
- w/o cutoff	$64 \times 96 \times 64$	0.6 / 2.4	0.18	36 / 101	0.5
- base only	$64 \times 96 \times 64$	0.2 / 0.8	0.18	1 / 6	0.5
- vortex part.	$64 \times 96 \times 64$	0.4 / 1.5	0.18	5 / 16	0.6
Street Fig.9	$40 \times 40 \times 128$	1.0 / 1.8	0.2	11 / 41	0.9
Duck Fig.10	$64 \times 96 \times 64$	0.8 / 3.1	0.2	8 / 30	0.4
- VIC 64	$64 \times 96 \times 64$	0.1 / 0.3	0.2	0.2 / 0.4	6 / 16
- VIC 256	$256 \times 384 \times 256$	0.8 / 3.8	"	4 / 11	156 / 350

Table 1: *Performance measurements for our simulation runs. Timings are mean runtime per frame. Two values with a " / " denote the mean and maximum values, respectively.* Grid *refers to all Eulerian operations, while* Mesh *represents vortex sheet dynamics. All simulations were run on a workstation with an Intel Core i7 CPU, a NVidia GTX 580 graphics card and 8GB of RAM.*

flow leads to complex density surfaces developing over time from the buoyant turbulence. Note that the irregular walls of the first, and the pulsed inflow of the second example would be difficult to realize with a simulation based on a pure vorticity formulation.

Wall turbulence In a next example, the interplay between mesh buoyancy and our turbulence model is investigated. To this end, we simulate a plume under the influence of buoyancy and a moving obstacle. Fig. 1 shows the orthogonality of the both models: with only the turbulence model activated, we observe detailed structures forming in the wake of the obstacle, while the rest of the flow remains laminar. Once the vortex sheet model is enabled, the mesh shows small-scale deformations with correct orientation due to buoyancy. We show that by combining the two models, we can benefit from both the accurate prediction of source regions by the turbulent energy model, as well as the anisotropic generation of the vortex sheet method. This example exhibits a large number of highly detailed swirls, many of them less than a fifth of a cell in diameter. These surface details are not smeared out despite moving along with the fast and turbulent velocities. Representing this detail during the course of a purely grid-based simulation would require a large amounts of memory, and corresponding amounts of computation for the advection step.

Performance The two most costly steps are applying the Gaussian kernel to the mesh, and integrating Eq. (8). Since these operations are simple and do not depend on neighborhood information, we evaluate them on the GPU. This leads to an average time of 10s per frame for the example scenes shown. The majority of this time is spent on the vortex sheet evaluation, i.e. the performance primarily depends on the number of triangles in the mesh. The number of triangles is in turn determined by two factors: the shot length, as triangle numbers typically increase during the course of a simulation, and the re-meshing resolution Δl. The parameter Δl can therefore be used as a means for fine-tuning detail versus performance. The performance numbers and statistics for all scenes can be found in Table 1, where *base only* refers to the plume simulation without a turbulence model.

To evaluate the performance of our approach compared to the Vortex-in-Cell (VIC) scheme used, e.g., in Stock et al. [2008], we have simulated the buoyancy only setup shown in Fig. 10. We measured computation times up to 19 times faster using our algorithm. We note that our VIC implementation uses OpenMP, but no GPU acceleration. We still think that this comparison is a good indicator of the complexity of the algorithms, despite the fact that both implementations are not optimized to their full extent. We found that VIC is non-trivial to port to the GPU, while our algorithm is easily realized in CUDA with a few short kernel functions.

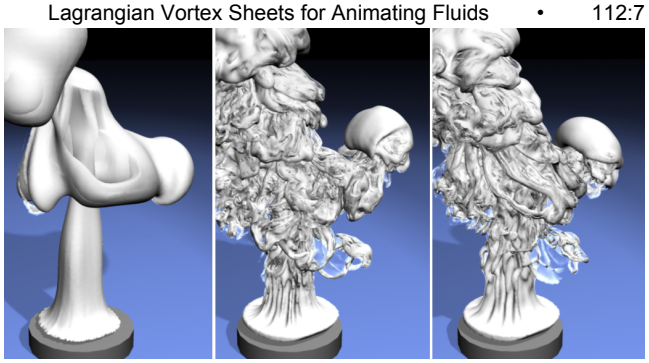

Figure 10: *We compare our method to Vortex-in-Cell integration. Our approach (middle) produces similar results as VIC on a 256 grid (right), while being 19 times faster. On the other hand, VIC with a resolution of 64 (left) has a comparable runtime to our method, but exhibits significantly less detail.*

8 Conclusion

We presented a novel algorithm for simulating buoyant, turbulent smoke plumes. We use a Lagrangian surface mesh to track the smoke/air interface. On this mesh, we solve the vortex sheet dynamics, and couple it to a low-resolution Eulerian fluid solver. This allows us to correctly simulate the turbulence generation process on the interface, which is important for visual coherency. On the other hand, the coupling with Eulerian large-scale dynamics allows us to evaluate the update of the velocity in a purely local fashion. This greatly reduces the complexity, and enables the efficient simulation of detailed plumes with non-trivial static boundaries or moving obstacles. In addition, we have proposed an orthogonal turbulence model for capturing turbulence production from obstacles.

A limitation of our approach is that it can lead to meshes with large numbers of triangles. Due to re-meshing, the number of triangles will often increase over time in turbulent regions for long simulation times. Although our resampling approach reduces the complexity of the meshes, more aggressive approaches are an interesting topic for future work. In addition, accumulated integration errors and re-meshing operations can lead to self-intersecting surfaces. We have, however, not encountered any problems when working with the resulting surfaces. Our method is naturally not well-suited for diffuse, hazy smoke. It would however be very interesting to combine our approach with a lower-resolution volumetric density representation. Sharp, detailed interfaces could then be tracked with our method, while the developing diffuse haze around the dense cloud could be represented on the volumetric grid. It would also be possible to add further detail based on the texture coordinates of the mesh, as we have a temporally coherent discretization of the surface over time.

Acknowledgments The authors would like to thank the reviewers for their comment and suggestions, and everyone at the CGL for the valuable discussions.

References

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

BARGTEIL, A. W., GOKTEKIN, T. G., O'BRIEN, J. F., AND STRAIN, J. A. 2006. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics 25*, 1.

BRADY, M., LEONARD, A., AND PULLIN, D. I. 1998. Regularized vortex sheet evolution in three dimensions. *J. Comput. Phys. 146*, 520–545.

BROCHU, T., AND BRIDSON, R. 2009. Animating smoke as a surface. *SCA posters*.

CHENTANEZ, N., AND MUELLER, M. 2011. Real-time eulerian water simulation using a restricted tall cell grid. *ACM Trans. Graph. 30*, 82:1–82:10.

CHORIN, A. J., AND BERNARD, P. S. 1973. Discretization of a vortex sheet on a roll-up. *J. Comp. Phys. 13*, 423–429.

COWPER, G. 1973. Gaussian quadrature formulas for triangles. *Int. J. Num. Methods 7*, 3, 405–408.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proc. SIGGRAPH*, 317–324.

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

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

KIM, D., SONG, O.-Y., AND KO, H.-S. 2009. Stretching and wiggling liquids. *ACM Transactions on Graphics 28*, 5, 120.

KOLLURI, R. 2005. Provably good moving least squares. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 1008–1018.

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.

LEONARD, A. 1980. Vortex methods for flow simulation. *J. Comput. Phys. 37*, 289–335.

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

LOZANO, A., GARCA-OLIVARES, A., AND DOPAZO, C. 1998. The instability growth leading to a liquid sheet breakup. *Phys. Fluids 10*, 9, 2188–2197.

MENG, J. C. S. 1978. The physics of vortex-ring evolution in a stratified and shearing environment. *J. Fluid Mech.*, 3, 455–469.

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. *ACM SIGGRAPH / EG Symposium on Computer Animation*.

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.

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.

PFAFF, T., THUEREY, N., COHEN, J., TARIQ, S., AND GROSS, M. 2010. Scalable fluid simulation using anisotropic turbulence particles. *SIGGRAPH Asia papers*, 174:1–174:8.

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*.

ROSENHEAD, L. 1931. The formation of vorticies from a surface of discontinuity. *Proc. Roy. Soc. London 134*, 170–192.

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. *Proceedings of ACM SIGGRAPH 24*, 3, 910–914.

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

SPALART, P. R., AND ALLMARAS, S. R. 1992. A one-equation turbulence model for aerodynamic flows. *AIAA Paper 92*, 0439.

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*.

STOCK, M., DAHM, W., AND TRYGGVASON, G. 2008. Impact of a vortex ring on a density interface using a regularized inviscid vortex sheet method. *J. Comp. Phys. 227*, 9021–9043.

TRYGGVASON, G., AND AREF, H. 1983. Numerical experiments on hele-shaw flow with a sharp interface. *J. Fluid Mech.*, 1–30.

WEISSMANN, S., AND PINKALL, U. 2010. Filament-based smoke with vortex shedding and variational reconnection. *ACM Transactions on Graphics 29*, 4.

WOJTAN, C., THUEREY, N., GROSS, M., AND TURK, G. 2010. Physics-inspired topology changes for thin fluid features. *ACM Transactions on Graphics 29,3* (July), 8.

WU, J.-Z. 1995. A theory of three-dimensional interfacial vorticity dynamics. *Phys. Fluids 7*, 10, 2375–2395.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *Proceedings of ACM SIGGRAPH 24*, 3, 965–972.

A Energy Transfer Model

The energy dynamics are described using the well-established k–ε model by Launder and Sharma [1974]

$$\begin{aligned} \frac{\mathrm{D}k}{\mathrm{d}t} &= \nabla(\frac{\nu_T}{\sigma_k}\nabla k) + \mathcal{P} - \varepsilon \\ \frac{\mathrm{D}\varepsilon}{\mathrm{d}t} &= \nabla(\frac{\nu_T}{\sigma_\varepsilon}\nabla \varepsilon) + \frac{\varepsilon}{k}(C_1\mathcal{P} - C_2\varepsilon) \quad , \end{aligned} \tag{11}$$

where $\mathcal{P}$, ε denote production and dissipation of turbulence, and $\nu_T = C_\mu \frac{k^2}{\varepsilon}$ is the turbulent viscosity. Launder and Sharma specify the model constants as $\sigma_k = 1$, $\sigma_\varepsilon = 1.3$, $C_1 = 1.44$, $C_2 = 1.92$ and $C_\mu = 0.09$. The k–ε model has inherent stability problems, especially due to k in the divisor of Eq. (11) which leads to instabilities for flows with low turbulence. We therefore ensure that k and ε are always in a meaningful range where a minimal amount of ambient turbulence is present. Bounds for k are given in terms of turbulence intensity I as $k = \frac{3}{2}{U_0}^2 I^2$, with the characteristic velocity U_0 which is an estimate of the velocity scale in the simulation. We use $I_{min} = 10^{-3}$, $I_{max} = 1$. We found ε is best limited using the equation for the turbulent viscosity ν_T, as this parameter linerly affects production. In our experiments, $\nu_{min} = 10^{-3}$, $\nu_{max} = 5$ are used. As starting parameters for a weakly turbulent initial state we found $\nu_T = 0.1$, $k = 0.1$ to produce stable results.