Simulation of Smoke Based on Vortex Filament Primitives

Alexis Angelidis, Fabrice Neyret

Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)
K. Anjyo, P. Faloutsos (Editors)

Simulation of Smoke based on Vortex Filament Primitives

Alexis Angelidis
University of Otago
alexis@cs.otago.ac.nz

Fabrice Neyret
GRAVIR/IMAG-INRIA
Fabrice.Neyret@imag.fr

Abstract

We describe a method that permits the high performance simulation of fluid phenomena such as smoke, with high-level control for the artist. Our key primitives are vortex filament and vortex ring: vorticity defines a flow as well as velocity does, and for numerous interesting flows such as smoke or explosions this information is very compact and tightly linked to the visual features of the fluid. We treat these vortices as 1D Lagrangian primitives (i.e. connected particles), which permit unbounded fluids and very accurate positioning of the features. The simulation of passive density particles for rendering is totally independent of the fluid animation itself. Thus, the animation can be efficiently simulated, edited and even stored, while the fluid resolution used for rendering can be arbitrarily high. We aim at plausible fluids rather than physical accuracy. For efficiency and stability, we introduce a new formalization of the Biot-Savart law and a modified Biot-Savart kernel. Our model also introduces a hierarchical filament structure for animation LOD, turbulent noise, and an original scheme for density particles.

1. Introduction

The interest of the Computer Graphics community for the simulation of gaseous phenomena has been growing over time. Various paths have been followed in order to better adapt solutions to the peculiar requirements of CG applications: Eulerian [KM90, FMb], Lagrangian [MP89], semi-Lagrangian [Sta99, FSJ01], spectral [Sta01], etc.

A common challenge is to obtain the **fastest computation time** for the **maximum possible fluid resolution**. Knowing that graphics applications tolerate trading accuracy for efficiency can help in choosing a scheme: e.g. the unconditional stability of [Sta99] permits using large time steps. Constraints due to the grids in Eulerian methods are released by [SCP*04]. Mixed models can increase the apparent resolution by relying on simpler models at small scales (carried by high level primitives), such as noise [Ney03, SSEH03] or procedural models [WH91], or by combining such high-resolution simple 3D models to interpolated 2D simulations [RNGF03].

A general problem especially important for CG is to obtain a **living fluid**: most methods suffer from numerical dissipation (intrinsic to Eulerians, and due to resampling for Lagrangians) in which small scale eddies die too quickly. To counter this, vorticity confinement was introduced in CG by [FSJ01], and sub-grid analytical models in [Ney03]. Alternatively an adaptive grid [LGF04] can be used.

Another challenge is to ease the **control of the fluid** by an artist. The high-level primitives of the mixed models mentioned above are naturally adapted for this. More recently, techniques have been proposed to target specific states of the fluid by controlling the whole field [FL, MTPS], or by controlling particles [REN*04, PCS04].

In this paper, we introduce a new path to CG fluids: simulation in the **3D vorticity space**[1]. The vorticity space is dual to the velocity space (see Section 2). But numerous fluid features appear more structured in vorticity space, as a multi-scale combination of *vortex filaments* (swirls, tornadoes) and *vortex rings* (smoke rings, explosion plumes, mushroom clouds). In numerous interesting situations the flow is characterized by a few such primitives, which are

[1] Note that this path has already been introduced to CG in 2D by [GLG95]. But 3D vorticity is very different since it is vectorial and highly spatially structured, see Section 2.

tightly connected to the visible features of the fluid: these primitives are thus interesting handles for user control. We represent them as 1D curves, i.e. connected particles. In the framework permitted by our model, the user can interactively create or modify such primitives. Procedural generation can also be used, e.g. to introduce turbulent fluctuations, as a more physical feature than the usual noise functions.

The velocity field can be reconstructed at any time from the vorticity filaments thanks to the Biot-Savart law (3), i.e. the animated flow is totally defined by a few animated curves. Thus, the motion can be simulated quickly. Moreover, these curves can easily be edited, replayed for tuning, keyframed, interpolated, or even stored, in the spirit of [PCS04]. Costly rendering can be done later at arbitrary resolution, and changing resolution will not modify the animation contrary to Eulerian or semi-Lagrangian methods (as mentioned in [LF02]).

Inconveniently, each vorticity element induces motion in the whole field so that computing the Biot-Savart integral (3) can be time consuming. Moreover, local self-induction can cause numerical instabilities. In this paper, we introduce a new formalization which induces a higher order scheme, permitting larger time steps. We also introduce an approximation which permits analytical integration and also stabilizes the simulation. Moreover, we propose a filament LOD scheme. Thus, our model allows us to efficiently compute the velocity induced at any given location by all the filaments.

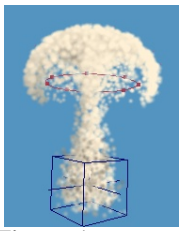


Figure 1: Flow induced by a vortex ring (*in red*). Smoke particles are generated in the box.

Section 2 reviews the concepts, equations and properties related to the vortical aspects of fluids. In Section 3 we revisit these equations in order to permit a higher order solver, and we detail our filament representation. Practical adaptations are proposed in Section 4. In Section 5, we describe the representation and the simulation of our vortex primitives, comprising an adaptive scheme, an LOD hierarchy and a noise function. Smoke particles are treated in Section 6. We present our interactive application in Section 7 and discuss results in Section 8.

2. The Physics of Vorticity and Filaments

2.1. The Lagrangian Vorticity Expression of Fluids

Vorticity-based approaches – called *Vortex methods* – are already used in Computational Fluid Dynamics [CK00]. They are especially adapted to turbulent fields and simulation of eddies since they track thin features nicely. Because the thickness of these features can be far smaller than any reasonable grid cell step, they are also more accurate [CMOV02]. As a Lagrangian approach, they do not suffer from the numerical dissipation which tends to kill small eddy structures when using Eulerian approaches.

The vorticity $\boldsymbol{\omega}$ is defined as $\mathrm{curl}(\mathbf{v})$, also denoted by $\nabla \times \mathbf{v}$ (where $\mathbf{v}$ is the velocity). Assuming incompressibility, mass conservation can be expressed as $\mathrm{div}(\mathbf{v}) = 0$. The Lagrangian formalism follows the properties of fluid parcels represented by *particles* advected along the flow with velocity $\mathbf{v}$. The Lagrangian formalism is especially adapted to vorticity since the non-zero vorticity is generally concentrated in loci (the vortices) which follow the flow.

In 2D, the Lagrangian vorticity form of the Navier-Stokes equation for inviscid fluids is simply $\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} = 0$. It means that once created, vorticity never dies and is simply advected along the field. Handling the 2D case is simple since it only requires vortices placed at isolated particles. It has been used in CG by [GLG95].

In 3D, the equation is:
$$\frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{v} \qquad (1)$$
It means that while following the flow, vortices are stretched by its local deformation. The 3D case is far more complicated since the vorticity is a vector and is spatially structured in filaments, which are often rings (i.e. closed loops). The *strength* of a filament is defined as the circulation Γ around the tube:
$$\Gamma = \int_L \mathbf{v} \cdot \mathrm{d}\mathbf{l} = \iint_S \boldsymbol{\omega} \cdot \mathrm{d}\mathbf{S} \qquad (2)$$
where S is a cross section of the tube and L the border line of S. This structure of vorticity has several consequences [Bat67, Mar97]:

• Firstly, since vorticity in a location induces a rotational motion everywhere in the fluid, parts of the same filament induce each-other: filaments self-induce deformations (e.g. oscillation modes) and global motion (e.g. a smoke ring moves straight on due to its self-induction). And of course, filaments interact with each other. As an example, two concentric rings will leapfrog through each other, i.e. one sucks the other which will suck the first right after, and so on.

• Secondly, filaments never die[2] and behave in a peculiar way when stretched: as stated by Kelvin's theorem [Rut89], the circulation Γ is constant both along the filament and in time, which means that the vorticity *increases* when the radius of the filament decreases due to the stretching! [3] This behavior can be thought of as the conservation of the angular momentum. As the fluid motion creates ubiquitous stretching, vortical areas quickly tend to concentrate into tubes, then to increasingly thinner filaments. This complex structure of turbulent fluids is what makes them so complicated to simulate, and explains why the classical methods lose important features.

Due to stretching, vortex tubes are often assumed to have a small core (hence the name *filament*). Thus, they can be conveniently represented by a 1D curve plus the circulation Γ

[2] As long as the inviscid hypothesis is valid. In practice filaments are dissipated when they become too thin.

[3] In particular, turbulence is made of a dense soup of very thin very rapidly rotating filaments.

rather than a very concentrated explicit $\boldsymbol{\omega}$ field. Since the circulation is preserved over time, no equation is needed for the evolution of the vortex strength.

Lagrangian primitives used in vortex methods can be 0D (regular particles), 1D (curves made of connected particles) or even 2D (since vorticity often starts as a 2D stretched sheet between two fluid layers before degenerating into vortex tubes then filaments[4]). Note that 0D particles [Gha01] lose the filament coherency, and thus have to explicitly track the effects of the stretching.

2.2. Reconstructing the Velocity Field

Recovering $\mathbf{v}$ from $\boldsymbol{\omega}$, i.e. obtaining $\mathbf{v} = \text{curl}^{-1}(\boldsymbol{\omega})$, is not easy. The solution at point $\mathbf{p}$ is given by the Biot-Savart law:

$$\mathbf{v}(\mathbf{p}) = \frac{1}{4\pi} \iiint_{\mathbf{x}} \frac{\boldsymbol{\omega}(\mathbf{x}) \times (\mathbf{p}-\mathbf{x})}{|\mathbf{p}-\mathbf{x}|^3} \, \mathrm{d}\mathbf{x} \tag{3}$$

Three comments can be made about this formula:

• Firstly, the solution of $\text{curl}^{-1}(\boldsymbol{\omega})$ is not unique: Equation (3) only provides a divergence-free solution, to which we can add any velocity field $\mathbf{v}_h$ satisfying both $\text{curl}(\mathbf{v}_h) = 0$ and the fluid hypotheses (mass conservation, boundary conditions). This harmonic field $\mathbf{v}_h$ corresponds to a solution of the flow using the simplest assumptions. In vorticity-based physical methods, it is solved separately. In our case, we will assume it is given by the user (or simply zero), so that in the following we do not handle it explicitly and we only consider the Biot-Savart solution.

• Secondly, evaluating Equation (3) at $\mathbf{p}$ using numerical integration is expensive, since it is performed with $\mathbf{x}$ over the entire space. To avoid this cost, *Vortex-In-Cell* methods rely instead on a finite difference solver on a grid to invert $\text{curl}(\boldsymbol{\omega})$, with the various drawbacks associated to grid sampling (comprising dissipation). We introduce analytical integration and LODs to avoid this cost.

• Thirdly, the integrand diverges at $\mathbf{p} = \mathbf{x}$, which corresponds to the evaluation of local self-induction. This can lead to a singularity[5], or at least to numerical instabilities. We will introduce a modified Biot-Savart kernel to avoid this.

2.3. Boundary Conditions

Incoming and outgoing flux are typically accounted for by the harmonic component of the velocity field (i.e. by solving with the divergence-free irrotational assumption). Interaction with objects is not easy for pure Lagrangian vortex methods (see section on future work). Nevertheless, some simple situations can be handled easily: it has been shown in [LNC91] that the interaction of a ring filament with a flat border with slip condition (i.e. only the normal component of velocity cancel) is equivalent to the interaction of the filament with its mirror image. Note that the no-slip condition can be obtained by inserting vorticity near the boundary so that the tangent component of the velocity cancel.

3. Our Choice of Representation and Solver

In Vortex Methods, the Biot-Savart law provides the velocity of every particle. We propose a new formalization of this law (detailed in Section 3.1), introducing a *whirl* operator which lets us recover higher order information about the trajectory of particles. This improves the precision of our solver (presented in Section 3.3) and therefore permits larger time steps.

3.1. Our Biot-Savart Reformulation

Let us consider a rotation of center $\mathbf{x}$, angle $|\boldsymbol{\omega}|$ and axis $\boldsymbol{\omega}$. The velocity of this rotation at a point $\mathbf{p}$ is $\boldsymbol{\omega} \times (\mathbf{p}-\mathbf{x})$, and can be represented by a 4×4 matrix, called *rotation velocity matrix*:

$$\mathbf{T}_{\boldsymbol{\omega}}^{\mathbf{x}\times\boldsymbol{\omega}} = \begin{pmatrix} 0 & -\boldsymbol{\omega}_z & \boldsymbol{\omega}_y & \mathbf{x}_y\boldsymbol{\omega}_z - \mathbf{x}_z\boldsymbol{\omega}_y \\ \boldsymbol{\omega}_z & 0 & -\boldsymbol{\omega}_x & \mathbf{x}_z\boldsymbol{\omega}_x - \mathbf{x}_x\boldsymbol{\omega}_z \\ -\boldsymbol{\omega}_y & \boldsymbol{\omega}_x & 0 & \mathbf{x}_x\boldsymbol{\omega}_y - \mathbf{x}_y\boldsymbol{\omega}_x \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{4}$$

This matrix is sparse and has only 6 degrees of freedom. It can be represented by two 3D vectors[6] $\boldsymbol{\omega}$ and $\mathbf{x} \times \boldsymbol{\omega}$. We introduce an operator $\mathbf{T}$ and denote by $\mathbf{T}_{\boldsymbol{\omega}}^{\mathbf{x}\times\boldsymbol{\omega}}$ the above matrix. Note that the operator $\mathbf{T}$ is linear:

$$\mathbf{T}_{\mathbf{a}_1+\mathbf{a}_2}^{\mathbf{b}_1+\mathbf{b}_2} = \mathbf{T}_{\mathbf{a}_1}^{\mathbf{b}_1} + \mathbf{T}_{\mathbf{a}_2}^{\mathbf{b}_2} \quad , \quad \mathbf{T}_{\alpha\mathbf{a}}^{\alpha\mathbf{b}} = \alpha\mathbf{T}_{\mathbf{a}}^{\mathbf{b}}$$

Let us call the scalar function $\beta_{\text{BS}}(\mathbf{x}) = \frac{1}{4\pi|\mathbf{x}|^3}$ the *Biot-Savart kernel*. The Biot-Savart law defines the velocity of a particle at $\mathbf{p}$ as the weighted sum of rotation velocity matrices:

$$\mathbf{v}(\mathbf{p}) = \left(\iiint_{\mathbf{x}} \beta_{\text{BS}}(\mathbf{p}-\mathbf{x}) \; \mathbf{T}_{\boldsymbol{\omega}}^{\mathbf{x}\times\boldsymbol{\omega}} \, \mathrm{d}\mathbf{x} \right) \cdot \mathbf{p} \tag{5}$$

We call the integrand in Equation (5) the *whirl of a vortex*, a weighted rotation velocity matrix:

$$\boxed{\varphi(\mathbf{p}, \mathbf{x}, \boldsymbol{\omega}) = \beta_{\text{BS}}(\mathbf{p}-\mathbf{x}) \; \mathbf{T}_{\boldsymbol{\omega}}^{\mathbf{x}\times\boldsymbol{\omega}}}$$

Note that φ is linear in $\boldsymbol{\omega}$: $\varphi(\mathbf{p}, \mathbf{x}, \boldsymbol{\omega}) = |\boldsymbol{\omega}| \varphi(\mathbf{p}, \mathbf{x}, \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|})$.

φ represents the velocity induced by an atomic *vortex element*. The angle of rotation is maximal at the vortex center $\mathbf{x}$. The scalar $\beta_{\text{BS}}(\mathbf{p}-\mathbf{x})$ describes how the rotation velocity decreases with distance.

[4] Known as the Kelvin-Helmholtz instability.

[5] And it does for several theoretical filament models, which makes their theoretical study so complicated [Mar97].

[6] The notation $\mathbf{x} \times \boldsymbol{\omega}$ represents the 3 top components of the right column, $(\mathbf{x}_y\boldsymbol{\omega}_z - \mathbf{x}_z\boldsymbol{\omega}_y \; , \; \mathbf{x}_z\boldsymbol{\omega}_x - \mathbf{x}_x\boldsymbol{\omega}_z \; , \; \mathbf{x}_x\boldsymbol{\omega}_y - \mathbf{x}_y\boldsymbol{\omega}_x)$. Note that this algebraic definition does not define a genuine geometric cross product since $\mathbf{x}$ is a point and not a vector. However, in practice it only appears in subtractions in the final equations, so that once factored the geometrical operation will really correspond to cross products of vectors.

We call the integral of φ over space the *whirl of a fluid*:

$$\Phi(\mathbf{p}) \quad = \quad \iiint_{\mathbf{x}} \varphi(\mathbf{p},\mathbf{x},\boldsymbol{\omega})\ \mathrm{d}\mathbf{x}$$

Thus the Biot-Savart expression becomes: $\mathbf{v}(\mathbf{p}) = \Phi(\mathbf{p}) \cdot \mathbf{p}$. Due to the linearity of $\mathbf{T}$ the matrix $\Phi(\mathbf{p})$ is as sparse as $\mathbf{T}$; thus only 6 scalar integrals have to be calculated to obtain it. We will see in Section 5 that we are able to lower this down to 3 or even 1 scalar integral in some situations. The matrix $\Phi(\mathbf{p})$ encodes a rigid body motion [7], i.e. a twist [Ang04].

3.2. Spatial Integral and Field Representation

$\Phi(\mathbf{p})$ has to be calculated at every point $\mathbf{p}$ to get the velocity induced by the vorticity field $\{\boldsymbol{\omega}(\mathbf{x}), \mathbf{x} \in \mathit{space}\}$. A numerical integration over the entire space would be very expensive.

We draw on the classical *vortex filament* assumption mentioned in Section 2.1: we consider that the vorticity is concentrated in thin tubes (i.e. filaments) C_i and null elsewhere[8]. The flow is thus entirely defined by the set $C = \{C_i, i \in [1,n]\}$.

The filaments are considered as differential elements, i.e. 1D curves with a formal radius $r(u)$. The vorticity is tangent to the curves. Various analytical profiles of the vorticity through a tube section are considered in the literature: e.g. constant or Gaussian. In fact, the only meaningful notion is the circulation Γ which is the integral of the vorticity on a section (see Equation (2)). Let us introduce the notation $\boldsymbol{\Gamma}$ with $|\boldsymbol{\Gamma}| = \Gamma$ and $\frac{\boldsymbol{\Gamma}}{|\boldsymbol{\Gamma}|} = \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|}$. We can call the expression $\varphi(\mathbf{p},\mathbf{x},\boldsymbol{\Gamma})$ the *whirl of a section* and forget about $\boldsymbol{\omega}$ and r. Thanks to Kelvin's theorem, Γ is constant along a filament and over time for inviscid fluids, even when considering stretching (see Section 2.1). As circulation represents the intuitive notion of the *strength* of a vortex filament, we consider it as a user-defined parameter.

However, we will see in Section 4.2 that we still need to store and maintain the filament thickness $r(u)$ – which decreases with stretching – if we want to take viscosity into account, since damping is highly dependent on r. Viscosity will affect the strength locally, thus we need to store $\boldsymbol{\Gamma}(u)$. Thus, each filament is defined by a parametric curve holding positions, circulation and radius:

$$\boxed{C_i = \{\{\mathbf{x}_i(u), \boldsymbol{\Gamma}_i(u), r_i(u)\}, u \in [0, L_i]\}}$$

We now simply have to compute 1D integrals representing the *whirl of the filaments*:

$$\boxed{\Phi(\mathbf{p},C) = \sum_i \int_0^{L_i} \varphi(\mathbf{p}, \mathbf{x}_i(u), \boldsymbol{\Gamma}_i(u))\ \mathrm{d}u} \qquad (6)$$

We handle the set of filaments C as a parameter of Φ since the filaments, i.e. the support of integration, move in time.

3.3. Time Integration Scheme

Let us consider a particle at location $\mathbf{p}$ in the fluid. The matrix $\Phi(\mathbf{p},C)$ in Equation (6) gives us access to the velocity at $\mathbf{p}$ through $\boxed{\mathbf{v}(\mathbf{p}) = \Phi(\mathbf{p}) \cdot \mathbf{p}}$, and thus to an estimate of the trajectory of $\mathbf{p}$ during the time step δt: $\tilde{\mathbf{p}}' = \mathbf{p} + \tau\ \mathbf{v}(\mathbf{p})\ ,\ \tau \in [0, \delta t]$. But the matrix Φ can provide more information than just the velocity. As we have already mentioned in Section 3.1, Φ encodes a *twist*, i.e. a rotation combined with a translation, whose matrix can be recovered with $\exp(\Phi)$ (see Appendix A). This provides us with higher order information about the trajectory of $\mathbf{p}$. Thus we compute the new location $\mathbf{p}'$ after a time step δt as:

$$\boxed{\mathbf{p}' = f(\mathbf{p}) = \exp(\delta t\ \Phi(\mathbf{p},C)) \cdot \mathbf{p}} \qquad (7)$$

Since $f(\mathbf{p})$ is of higher order than a translation, the estimate $\mathbf{p}'$ is more accurate than $\tilde{\mathbf{p}}'$. This allows us to make larger time steps and therefore to gain speed. Moreover, if the flow is a pure rotation, translation or twist, the reconstructed trajectory of $\mathbf{p}$ will follow it exactly regardless of the length of time step δt.

Note that the two first terms of the series expansion of exp provide the linear trajectory $(\mathrm{I} + \delta t \Phi) \cdot \mathbf{p}$, so our scheme is asymptotically equivalent to a simple Euler integration step.

We use the scheme based on Equations (6) and (7) for animating marker particles in the fluid as well as the points $\mathbf{x}_i$ defining the filaments. The evolution of the set of filaments C after a time step δt is thus simply defined by $\boxed{C' = f(C)}$. This is simply a restatement of Equation (1).

4. Practical Approximations and Extensions

To gain even more efficiency, we want to avoid costly numerical computation of the Biot-Savart integrals along the vortex primitives that will be defined in Section 5. For this, we look for closed-forms. Our strategy is to replace the Biot-Savart kernel with another (see Section 4.1) which eases analytical integration and which is more stable.

At this stage we will have an engine for incompressible inviscid fluid in an unbounded space. We show how viscosity and boundary conditions can be introduced in Section 4.2.

4.1. Changing the Biot-Savart Kernel

The Biot-Savart kernel β_{BS} has two drawbacks: it diverges at $\mathbf{0}$ ($\beta_{\mathrm{BS}}(\mathbf{0}) = \infty$), leading to numerical instabilities for particles that are very close to a vortex center (typically, the neighbor nodes on the filament), and it generally disables closed-forms for integrals.

We propose to replace the Biot-Savart kernel β_{BS} with another radial basis function β_{MS} which is defined and smooth around $\mathbf{0}$ and which eases the analytical integration:

$$\beta_{\mathrm{MS}}(\mathbf{x}) = \frac{1}{\pi(1 + |\mathbf{x}|^2/s^2)^2} \qquad (8)$$

[7] Formally, $\Phi(\mathbf{p})$ is an element of the Lie algebra $\mathfrak{se}(3)$, and the corresponding element $\exp(\Phi(\mathbf{p}))$ of the Lie Group $\mathbf{SE}(3)$ is a twist transformation [Gal02] (see Appendix A)

[8] Note that $\boldsymbol{\omega} = 0$ at some location does not mean that there is no motion there, since vorticity induces motion at a distance.

This kernel β_{MS} is proportional to the one introduced in [MS98] in the context of convolution surfaces. The coefficient s is a user controllable parameter related to the apparent thickness of the filament: the region closer than s to the curve tends to rotate like a solid core. During simulation s should be roughly proportional to r, but for stability it should not decrease below a threshold s_1. Thus we model $s(r)$ as $s_0 r + s_1 k(r)$ where $k(r)$ is a function decreasing from 1 to 0 and s_0, s_1 are such that $s(r)$ is monotonic. We chose $s_0 = 1$, $s_1 = \frac{2}{3}$ and $k(r) = e^{-\frac{3}{2}r}$.

This kernel smooths the local self-induction, but it also slightly overestimates the induction on distant particles. The resulting animation is still visually satisfactory. Note that changing the kernel β does not alter the incompressibility property; for instance, a piecewise polynomial kernel is also used in the context of volume preserving shape modeling [ACWK04].

4.2. Viscosity, Stretching and Boundaries

Viscosity: It has two effects on fluids. Firstly, it diffuses quantities (velocity, vorticity and markers). As often done, in this paper we consider that this effect is negligible at visible scales by assuming the fluid is inviscid, which yields the simple equations we use. Secondly, it prevents filaments from becoming infinitely thin with infinite vorticity, which makes real fluids free of singularities. This is also the very mechanism which *dissipates* the vortical energy transmitted[9] from higher scales. This effect occurs at very small scales, but it is important to take it into account in order to avoid singularities, endless accumulation of filaments and infinite growth of energy. We model this as a radius-dependent damping of the filament strength, done at each time step:

$$\Gamma_i'(u) = (1 - \nu(r_i(u)))^{\delta t}\, \Gamma_i(u) \qquad (9)$$

where $\nu(r)$ is a damping function decreasing from 1 to 0 with a characteristic viscous scale r_0. In our implementation we use $\nu(r) = e^{-\frac{r}{r_0}}$.

Weak filaments are faded out to zero and then destroyed. Note that independently from this physical decay, it is useful to allow the user to decide when to fade and kill a filament as mentioned in Section 7.

Radius Stretching: In order to know $r_i(u)$, we need to compute the vortex stretching during the animation of the filament. Let λ be the lengthening rate measured at a given filament location. The volume conservation of a small cylindrical portion of filament tells us that when its length multiplies by λ, its radius divides by $\sqrt{\lambda}$. Thus, we simply compute at each time step: $r_i'(u) = \frac{r_i(u)}{\sqrt{\lambda}}$.

Boundary Conditions: In this paper, we only deal with flat motionless boundaries with a slip condition. As explained in Section 2.3, in the vortex formalism one has simply to simulate the interaction of filaments with their mirror image through the border plane. Thus, we need to compute the whirl $\Phi'(\mathbf{p})$ of the mirrored filaments and its influence on a given point $\mathbf{p}$. Conveniently, it is equivalent to compute the whirl $\Phi(\mathbf{p}')$ of the regular flow at point $\mathbf{p}'$ which is the mirror of $\mathbf{p}$ relative to the plane. Let us denote by S the mirroring operator relative to the plane (i.e. $\mathbf{p}' = S \cdot \mathbf{p}$). Then we have $\Phi'(\mathbf{p}) = S \cdot \Phi(S \cdot \mathbf{p}) \cdot S$. The total flow is simply $\Phi(\mathbf{p}) + \Phi'(\mathbf{p})$. It can be shown that if $\mathbf{p}$ is on the plane, then $\exp(\Phi + S \cdot \Phi \cdot S)$ is a translation tangent to the plane.

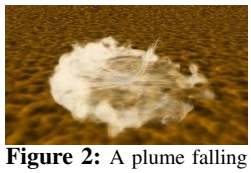


Figure 2: A plume falling on the floor.

In practice, we only consider Φ' for filaments and $\mathbf{p}$ close enough to a boundary. Moreover, we extend this case for moderately curved boundaries with a tangent plane approximation.

5. Our Primitives of Vorticity

The whirl Φ of a fluid is defined by Equation (6) as the sum of the whirl of each filament. The purpose of this section is to describe how we represent the filaments and how we compute their whirl, taking advantage of the adapted kernel β_{MS} defined in Equation (8). As we have seen in Section 3.3, from this whirl we can compute the displacement and the velocity at every point $\mathbf{p}$ in the flow. This is used to advect all the particles, comprising the filaments.

Every vortex element in the flow influences every particle. To save computations, we introduce a hierarchy of models to represent filaments, and LODs for the finest model.

- The finest filament model consists of a set of connected particles. The computation of the filament whirl is based on the integration of φ on its segments. We detail it in Section 5.2, as well as its LOD structure.
- The coarser level consists of a circular ring, treated in Section 5.1. A circle is an approximation which makes sense since flow perturbations often start as simple vorticity rings, which can remain circular for a while depending on the environment. It also makes sense to approximate small rings by circles since the extra detail would have little effect at distance. Conveniently, the whirl of a circle can be computed analytically which makes it especially efficient.
- Similarly, a coarse model should be handled for straight filaments. In fact, this case can be handled directly as the coarsest LOD level of the regular filament model.
- For the coarsest level we introduce a vortex noise model consisting of isolated vortex primitives. We detail it in Section 5.3.

For each of these models we describe how to evaluate their whirl and how to update their structure through simulation.

5.1. Circular Ring

Circular Ring Whirl: A circle is defined at each time by a center $\mathbf{c}$, a radius k, and a vector $\mathbf{z}$ perpendicular to the

[9] Through the Kolmogorov cascade.

circle plane. The symbolic integration of the section whirl along the circle gives the following whirl:

$$\Phi_{\text{circle}}(\mathbf{p}) \;=\; k\int_u \varphi(\mathbf{p},u)\;\mathrm{d}u \;=\; \Gamma\, b\, \mathbf{T}_{\eta}^{\mathbf{c}\times\eta+ak\mathbf{z}} \quad (10)$$

where $a = |\mathbf{p}-\mathbf{c}|^2+s^2+k^2$

$\eta = 2\mathbf{z}\times(\mathbf{p}-\mathbf{c})$ $\quad b = \frac{2s^4k}{\pi(a^2-k^2|\eta|^2)^{3/2}}$

Circular Ring Advection: The advection of our circular ring is done in three steps: taking samples on that circle, advecting the samples, and fitting a circle to the newly obtained positions. When the circle fitting error is too high according to a user-defined criterion, it can be swapped with a closed deformable filament.

5.2. Deformable Filament

A deformable filament C_i is represented with a polygonal curve, i.e. vertices connected by segments. The filament is simply deformed by advecting the vertices. The total whirl $\Phi_i(\mathbf{p},C_i)$, generated at a point $\mathbf{p}$ by the polygonal curve, is computed by summing the whirls generated by each segment. In the next subsection we describe how to compute the whirl of a segment. The polygonal curve to be used is determined according to our LOD scheme and an error criterion, which are described in the two following subsections. The result in $\mathbf{p}$ is reasonably valid for a neighborhood around $\mathbf{p}$, and thus the LOD to be used can be computed only once for a cluster of many particles. These clusters are defined using a floating grid which adapts to clouds of particles (typically, the smoke particles described in Section 6).

For defining the LODs hierarchy of a deformable filament we build a binary-tree of polygonal curves, whose nodes are segments. The leaves represent the segments of the detailed filament, and each internal node represents a segment which is the *average* of its two children.

Segment Whirl: Let us define a segment $(\mathbf{p}_0,\mathbf{p}_1)$ parameterized in $u\in[0,1]$. Let us denote by l the length of the segment. The symbolic integration of the section whirl along the segment gives:

$$\Phi_{\text{segment}}(\mathbf{p}) \;=\; l\int \varphi(\mathbf{p},u)\;\mathrm{d}u \;=\; \Gamma\, h(\mathbf{p})\, \mathbf{T}_{\mathbf{p}_1-\mathbf{p}_0}^{\mathbf{p}_0\times\mathbf{p}_1} \quad (11)$$

where the scalar function $h=\int \beta_{\text{MS}}(\mathbf{p},u)\,du$ is:

$$h(\mathbf{p}) = \frac{s^4}{2a^2\pi^2}\left(\frac{a_0}{d_0+s^2}+\frac{a_1}{d_1+s^2}+\frac{l^2}{a}(\arctan\frac{a_0}{a}+\arctan\frac{a_1}{a})\right)$$

in which
$$d_0=|\mathbf{p}-\mathbf{p}_0|^2 \qquad d_1=|\mathbf{p}-\mathbf{p}_1|^2$$
$$d=\tfrac{1}{2}(d_0+d_1-l^2) \qquad a^2=d_0d_1-d^2+l^2s^2$$
$$a_0=d_0-d \qquad a_1=d_1-d$$

Since a repeated evaluation of the above expression is expensive, an accurate approximation is useful. If we denote by $\mathbf{p}_{\min}$ and $\mathbf{p}_{\max}$ the closest and farthest points from point $\mathbf{p}$ on the segment, we can minimize and maximize terms in the integrand of Equation (11): if $|\beta_{\text{MS}}(\mathbf{p}_{\min})-\beta_{\text{MS}}(\mathbf{p}_{\max})|<\varepsilon$, then the following is a good approximation:

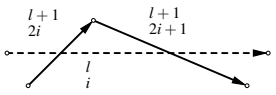
$$\begin{aligned}\tilde{\Phi}_{\text{segment}}(\mathbf{p}) &= \Gamma\,\tilde{h}(\mathbf{p})\,\mathbf{T}_{\mathbf{p}_1-\mathbf{p}_0}^{\mathbf{p}_0\times\mathbf{p}_1}\\ \text{where}\quad \tilde{h}(\mathbf{p}) &= \tfrac{1}{2}\left(\beta_{\text{MS}}(\mathbf{p}_{\min})+\beta_{\text{MS}}(\mathbf{p}_{\max})\right)\end{aligned} \quad (12)$$

Building the LOD Tree: The filaments deform during the simulation, so their LOD tree has to be reconstructed at each time step. To build the tree bottom-up, all that is required is a method for averaging pairs of neighbor segments.

Our criterion is to best preserve the whirl, i.e. that the whirl of each level of the tree is as close as possible to the whirl of the levels below, for any point where it will be evaluated later.

Finding a polygonal curve whose whirl best matches the whirl of a polygonal curve with twice as many segments is an expensive minimization problem, that we cannot afford to solve interactively. We propose the following simple scheme, which works well. Other schemes could be used, such as an inverse-subdivision scheme [SNBW03].

- The starting point is a detailed filament with 2^l segments. These correspond to the tree leaves.
- For each pair of neighbor segments $\{2i,2i{+}1\}$, we define the parent segment i with a length equal to the sum of the lengths of its children, and intersecting the children at mid-length. For its circulation, we simply take the average.



- We repeat this step until the root is reached, i.e. a single segment for open filaments, and at least three segments for closed filaments.

Choosing the LOD of a Whirl: Determining the LOD of the whirl to be evaluated for a point $\mathbf{p}$ (and its neighborhood) is done top-down by fetching finer segments in an adaptive non-uniform manner. The segment subdivision criterion is based on an estimate of the error produced when using the whirl of a single segment Φ_{e_0} instead of the sum of the whirls of its two children Φ_{e_1} and Φ_{e_2}. The exact geometric error is the distance between the transform of $\mathbf{p}$ by the twist encoded by Φ_{e_0} and the transform of $\mathbf{p}$ by the twist encoded by $\Phi_{e_1}+\Phi_{e_2}$ (applying Equation (7)):

$$\varepsilon(\mathbf{p}) = |\ \exp(\delta t(\Phi_{e_1}+\Phi_{e_2}))\cdot\mathbf{p}-\exp(\delta t\Phi_{e_0})\cdot\mathbf{p}\ |$$

In order to save costly computations, we rely on two approximations to estimate this error. Firstly, we approximate twists with translations, i.e. a first order approximation of the exponential: $\exp(M)\approx \mathrm{I}+M$. Secondly, for computing the matrices Φ_{e_i} we estimate the costly integration of Equation (11) by bounding the kernel β_{MS} for each segment. Thus, an approximation of the error is:

$$\tilde{\varepsilon}(\mathbf{p}) = \max_{ijk}\left|\beta_{\text{MS}}(\mathbf{q}_1^j)M_{e_1}\cdot\mathbf{p}+\beta_{\text{MS}}(\mathbf{q}_2^k)M_{e_2}\cdot\mathbf{p}-\beta_{\text{MS}}(\mathbf{q}_0^i)M_{e_0}\cdot\mathbf{p}\right|^2$$

where

M_{e_i} is the matrix $\delta t\ \Gamma_i\ \mathbf{T}_{\mathbf{p}_1+\mathbf{p}_0}^{\mathbf{p}_0\times\mathbf{p}_1}$, associated with edge i

$\mathbf{q}_i^0,\mathbf{q}_i^1$ are the closest and farthest points from $\mathbf{p}$ on e_i

Deformable Filament Advection: The leaf vertices of a filament are simply advected like particles, and the binary-tree is updated at each time step. Whenever the

leaf segments themselves are too stretched and become undersampled, several solutions are available:

- Add an extra LOD level by splitting all the segments.
- Resample the curve evenly.
- Wait for the filament to naturally vanish, since the over-stretching weakens it (see Section 4.2).
- Let the user decide when to fade out the curve, e.g. by keyframing $\nu(t)$.

5.3. Noise Vortices

The amount of detail that can be simulated with CFD methods is limited, since an increase of resolution requires a significant increase of computing resources. In order to circumvent this limitation, tricks can be used in CG for amplifying realism: various kinds of noise functions have been proposed in the literature, such as Perlin Noise [Per85], flownoise [PN01] and stochastic divergence-free fields [SF93, RNGF03]. But their visual quality suffers from the fact that the noise does not satisfy the fluid properties: only the last kind is divergence-free, and all of them lack the temporal coherency of eddies. In our formalism, a user can model an efficient and high quality noise by spawning *noise vortices* in areas where turbulence is wanted. A noise-vortex consists of a position $\mathbf{c}_i$, an axis of rotation $\mathbf{a}_i$ and a rotation amplitude Γ_i. It only influences marker particles, within a radius of influence r_i. It is advected in the flow like the other particles.

Figure 3: *Left:* Two leapfrogging circle filament and noise particles. *Middle:* Flow simulated without the noise. *Right:* With the noise particles.

An advected axis cannot be simply transformed using the Jacobian of the displacement $J(f)$ (where f is defined in Equation (7)) like a material tangent would: the stretching of the flow would tend to align neighboring axes along the main axis of the local stretch. In order to keep unorganized noise axes, the eigenvectors of J could be used; but cases arise where they are undetermined. We propose a simple scheme in which these eigenvectors are attractors (when they exist): we attach a sort of local Frenet frame $(\mathbf{t},\mathbf{n},\mathbf{b})$ to particles, composed of *tangent*, *normal* and *binormal* axes, updated as follows:

$$\mathbf{t}' = J\cdot\mathbf{t} \qquad \mathbf{n}' = J^c\cdot\mathbf{n} \qquad \mathbf{b}' = \mathbf{t}\times\mathbf{n}$$

where J^c is the cofactors matrix[10] of J (see [Bar] for justification). Then we define noise vortices among three categories, *tangent-vortices*, *normal-vortices* and *binormal-vortices*, whose rotation axes are defined by one of the frame axes.

[10] If we denote by $\{\mathbf{j}_0,\mathbf{j}_1,\mathbf{j}_2\}$ the columns of J, then the columns of J^c are $\{\mathbf{j}_1\times\mathbf{j}_2,\mathbf{j}_2\times\mathbf{j}_0,\mathbf{j}_0\times\mathbf{j}_1\}$.

6. Smoke Particles

In CG applications the visual fluid features consist of interfaces (water surface), the distribution of markers (smoke, cloud droplets, colors), or the advection of objects (e.g. leaves). Advection of passive[11] objects is done easily with our method by simply evaluating the whirl at the object location: this provides the new object position as well as its rotation.

Figure 4: Without (*left*) and with (*right*) particle distortion.

The purpose of this section is to describe our representation of marker densities. Eulerian methods can either treat this density[12] as an extra quantity to be updated at grid nodes, or rely on particles spawned in the simulated flow [FMa]. Naturally, Lagrangian methods rely on particles, i.e. floating markers whose position $\mathbf{p}_i$ is carried by the flow. Sizeless particles make it difficult to maintain a correct sampling of the visible features through simulation, and complex heuristics must be provided to generate new particles in undersampled dense areas. This can result in visual artifacts, especially for highly stretched flows. Instead, we consider *blob* particles [SF93] to which a reference size s_i and a density ρ_i are associated.

The fluid parcel corresponding to this volume will distort in a complicated manner through time. [SF95] reproduced this effect on large blobs using backwarped rays, but this technique does not easily apply to real-time rendering.

Assuming that particles are small and that the magnitude of their strain is tiny compared to the large scale motion of the fluid, we handle linear anisotropic distortion of blobs, i.e. we simulate *ellipsoid* blobs. This enables long smooth particles, which gives a high quality result at low cost (see Figure 4). When the stretching becomes too high for the linear assumption, we split the particle.

The ellipsoid shape of our blobs is represented by a quadratic form Q_i, whose eigenvectors $\{\mathbf{e}_0,\mathbf{e}_1,\mathbf{e}_2\}$ and eigenvalues $\{\lambda_0,\lambda_1,\lambda_2\}$ give the ellipsoid principal axes and squared radii. They can be recovered by diagonalizing the matrix Q_i.

$$Q_i = \begin{pmatrix}\mathbf{e}_0 & \mathbf{e}_1 & \mathbf{e}_2\end{pmatrix}\begin{pmatrix}\lambda_0 & 0 & 0\\ 0 & \lambda_1 & 0\\ 0 & 0 & \lambda_2\end{pmatrix}\begin{pmatrix}\mathbf{e}_0 & \mathbf{e}_1 & \mathbf{e}_2\end{pmatrix}^{-\mathrm{T}}$$

Stretching Smoke Particles: The strain added during a time step is given by the Jacobian[13] of the displacement $J=\nabla(f)$ where f is defined in Equation (7).

[11] Cross-interaction between large objects and the fluid is a complex problem which is beyond the scope of this paper.

[12] It is important to note that this is not the *fluid* density.

[13] Note that [Ney03] only considers the norm of the strain and thus does not capture the directional information.

The ellipsoid shape of our blobs directly represents the accumulated distortion. Starting with $Q_i = s_i \mathrm{I}$, at each time step we compute how the ellipsoid is deformed with $Q_i' = J(\mathbf{p}) \cdot Q_i \cdot J^{\mathrm{T}}(\mathbf{p})$. Note that aside from numerical errors, incompressibility yields $det(J) = 1$ so the volume of the blob is preserved.

Splitting Smoke Particles: Let us denote by λ_0 the largest eigenvalue of the deformation tensor Q_i, corresponding to the main axis $\mathbf{e}_0$ of the ellipsoid. When $\sqrt{\lambda_0}/s_i$ exceeds a threshold the particle is too stretched, so it is split across the stretching direction $\mathbf{e}_0$. Two children particles are generated in place of the parent particle with the same axes and radii $(\sqrt{\lambda_0}/2, \sqrt{\lambda_1}, \sqrt{\lambda_2})$. Each child keeps the same reference size s_i, inherits half of the density ρ_i, and is placed at point $\mathbf{p}_i \pm \frac{\sqrt{\lambda_0}}{2}\mathbf{e}_0$ ($\mathbf{e}_0$ is assumed to be unitary).

Drawing Particles: An ellipsoid particle is easy to render since its projection in screen space can be obtained analytically. Given two orthogonal 3D unit vectors $\mathbf{x}$, $\mathbf{y}$ contained in the viewing plane, the 2×2 projected matrix Q_{2D} is:

$$Q_{2D} = \begin{pmatrix} \mathbf{x}^{\mathrm{T}} \\ \mathbf{y}^{\mathrm{T}} \end{pmatrix} Q_i \begin{pmatrix} \mathbf{x} & \mathbf{y} \end{pmatrix}$$

Thus, to render the particles we sort the ellipsoids in depth and simply render each 2D ellipse on a billboard facing the camera: the orientation and size of the billboard are determined by Q_{2D} eigenvectors and square roots of eigenvalues. We only need a small texture containing a circular gradient of opacity, which is shared by all the splatted particles.

7. Interactive Design of Flows

For our tests we have implemented an interactive editor allowing users to specify and edit a flow. While simple enough, it illustrates how our representation allows a user to design, edit and control a flow.

Geometric objects are of two kinds: vortex filaments and smoke particles (plus obstacles). Vortex filaments consist of curves that can be interactively inserted in the scene, and then loaded or saved on disk. Smoke particles are treated similarly as in particle systems editors: their initial position is spread interactively or procedurally within simple volumes or on surfaces. Both types of objects have various associated attributes controlling their behavior or their appearance.

Figure 5: A vortex ring following a curve.

The framework of our scene editor is similar to the one of a classical CG animation system: the user can select the current time with a slider, then add or edit the geometrical content, tune the attributes, and keyframe geometrical or attribute data. When playing part of an animation in the editor, the keyframed data is treated in a standard way, while the non-keyframed data is simulated in real-time with our fluid engine. Both kinds are thus naturally integrated. Combinations are easy to manage: the user can keyframe the extinction of a simulated filament, or let a filament interpolate between a simulation and a keyframed curve, or switch from one mode to the other for a period of time.

At any time, the flow can be rendered either in fast or high quality rendering. The quality/efficiency ratio can be controlled in two ways. Firstly, by selecting the visual effects: e.g. shadows, complex lighting. Secondly, by tuning the smoke particles' global attributes: sampling density, self-subdivision enabling, ellipsoidal distortion enabling, and associated thresholds. As shown in the results, reasonable renderings can be obtained in real-time.

Thus, unlike usual fluid simulators whose various limitations from the CG point of view are mentioned in [LF02], the framework in our fluid editor is similar to that of a geometric modeler. In particular, the flow features are represented as *vectorial compact data*. This has several consequences:

- It is easy to store the entire animated scene and to edit it interactively, going back in time to change a detail.
- Features are meaningful and easy to handle for the artist.
- The simulation is resolution-independent, and deterministic in practice, as opposed to grid-based fluid simulations where results change when the grid size changes.
- It is easy to play several minutes of animation before reaching the relevant time range to be rendered.
- It is easy to re-render a given simulation with new rendering attributes, or to add new frames later.

8. Results

Features: The effects of vortex-induced motion, noise, collision on the floor, distortion of smoke particles, and keyframed vortices, are illustrated in Figures 1 to 5. The accompanying video presents these effects animated plus various examples from a simple plume to complex fields. Some are reproduced in Figure 6. The smoke sheet in a 3D flow (Figure 6c) deserves some comments: as is done in real-world wind tunnels, we have placed sources of smoke such that a thin sheet of markers interacts with the 3D flow.

Complexity and Performances: Let n_f be the number of filaments, k_f the total number of filament segments at the finest level, $\tilde{k}_f$ the average number of filament segments considered taking LOD into account, n_n the number of noise particles, n_s the number of smoke particles, $\tilde{n}_n$ the average number of noise particles acting on a smoke particle.

The **simulation cost** can be estimated from the number of evaluations of $\Phi_{segment}$ as $(k_f + n_n + n_s)\tilde{k}_f + n_s\tilde{n}_n$. Its most significant component is $n_s\tilde{k}_f$. This means that the simulation of filaments alone is almost free, which makes the interactive modeling of flow features easy. All the simulation time is spent on smoke particles. Accounting for particles distortion multiplies the cost by 4 due to the finite-difference estimation of the Jacobian. The grid LOD factorization yields a 15% saving. See below for possible improvements.

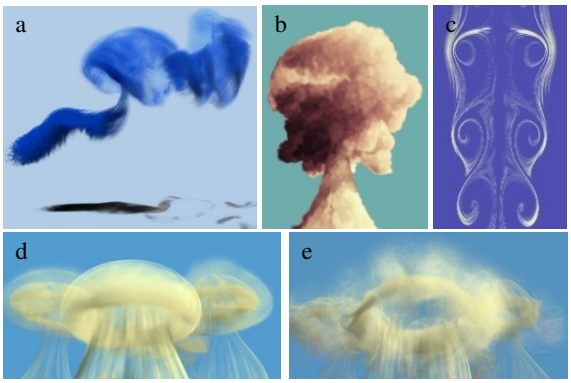


Figure 6: Various examples of animated flows.

The **rendering cost** decomposes into the splatting of smoke particles (comprising the calculus of the ellipse shapes) and the shadow calculation (self and cast). The shadows represent the main part of the rendering cost. We measured that the rendering cost was roughly independent of the resolution (about 2% of overhead for 12 times more pixels).

Benchmarks: The following performance measurements were done on a Pentium 4 processor at 2.8 GHz with an nVIDIA GeForce FX 5200 graphics board. Note that we measure simulation and rendering with smoke particles. Simulation and visualization of filaments alone in the flow editor is real-time.

• *Heavy explosion* (Figure 6b). The field is defined by 2 circle rings and 100 noise vortices. The smoke consists of about 5,000 non-deformable particles. The animation is rendered (without shadows) at 16 fps.

• *Smoke sheet* (Figure 6c). The field is defined by 10 circle rings. The smoke consists of about 30,000 deformable particles. The animation is rendered at 0.7 fps (or 3 fps using simple particles).

• *Train smoke* (Figure 6a). The field is defined by 20 circle rings and 20 filaments made of 64 segments each. The smoke consists of about 30,000 deformable particles. The animation is rendered (with shadows) at 12 seconds per frame, 29% of which is due to shadows, and 70% to advection ($\frac{1}{4}$) and distortion ($\frac{3}{4}$) of smoke particles.

• *Field of plumes* (Figure 6d and e). The field is defined by 6 filaments starting as circles then turning to deformable filaments made of 64 segments. 96 noise particles per filament were used for *e*, and no noise for *d*. The smoke consists of about 50,000 particles. The animation is rendered at 7 seconds per frame for *d* and 23 seconds per frame for *e*.

• *Comparison with [FSJ01]*, a reference for smoke simulation. It is not really possible to compare fairly an Eulerian and a Lagrangian method, since their respective resolution does not measure at all with the same standards. Moreover, what is an easy case for one method corresponds to the difficult case for the other and conversely (e.g. resolved detail vs crowded volume). Still, we tried to produce two flows where apparent complexity was roughly comparable to examples of [FSJ01]. In the following we have upgraded their timings according to our CPU clock.

- The field shown on Figure 7(left) is simulated at 0.9 fps. It resembles their Figure 3 which would play at 0.1 fps.
- The field shown on Figure 7(right) is simulated at 24 fps. It resembles their Figure 8 which would play at 1.6 fps.

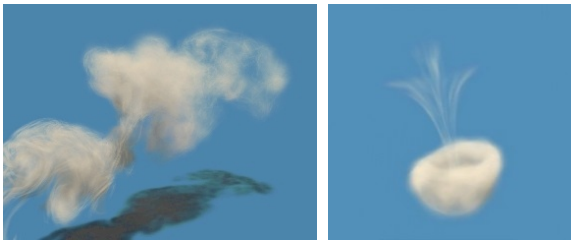


Figure 7: Benchmarking with scenes close to [FSJ01] Fig.3 and Fig.8.

Possible Enhancements to Improve Performances:

• As we have seen, deformation of smoke particles is very costly. The deformation of a particle could probably be estimated only once in a while. Moreover, the Jacobian could be calculated analytically rather than requiring 3 extra evaluations of Φ for each particle.

• Φ is evaluated billions of times. The floating grid only saves LOD estimations. It should be possible to save a lot more by interpolating in grid cells the components of Φ corresponding to distant vortices. In our formalism the results of the integration Φ is a whirl *operator* and not directly a new point or velocity, so a good quality interpolation can be expected.

• Smoke particles keep splitting with stretching, thus numerous diluted particles tend to appear. In our application we cancel particles under a density threshold, but this can lead to visual artifacts (vanishing smoke) since the accumulation of numerous diluted particles may still be visible. Neighboring diluted particles should be resampled and combined into bigger particles.

9. Conclusion and Future Work

We have presented a method which allows the fast and easy design and simulation of flows such as turbulent smoke by relying on a compact high-level primitive, the *vortex filament*, which induces the velocity field. As a geometric object, it is easy to edit and animate in a modeler. We have also presented a rendering scheme based on deformable particles to represent and render the smoke advected in this field. Our Lagrangian vorticity scheme does not suffer numerical dissipation and is not bounded by any grid. The simulation is independent from the rendering, whose resolution can be chosen and changed afterwards without side-effects on the simulation. Our animated examples show that very detailed results can be generated efficiently.

The issues faced by Vortex Methods in fluid engineering are also of interest for CG, even if in our domain we can circumvent most of the constraints. These issues concern complex environments and long simulations. For the former,

complex boundary conditions should be considered, and LOD should be extended to account for clusters of filaments. For the latter, the complex interaction on filaments should be modeled, especially their reconnections and collapses. Both issues are difficult to deal with using pure a Lagrangian approach. Possible solutions consist in mixing the approach with a grid-method for handling these. We may be not so far from interactive walks through detailed living features inside large flows!

Acknowledgments: Many thanks to Sui-Ling Ming-Wong and Chuck Hansen for proof-reading this article, Brendan McCane for great discussions on particles, and Geoff Wyvill for discussions. This research was supported by the Marsden Fund.

Appendix A: Exponential of Matrices

In this section, we describe how to compute the exponential of matrices of the form $M = \mathbf{T}^{\mathbf{m}}_{\boldsymbol{\omega}}$ as defined in Equation (4). The following formulas were obtained using the series definition of the exponential, $\exp M = \sum_{k=0}^{\infty} \frac{1}{k!} M^k$. Note that they are exact, i.e. not a third order expansion.

$$\exp M = \begin{cases} \mathrm{I} + M & \text{if } |\boldsymbol{\omega}| = 0 \\ \mathrm{I} + \frac{1-\cos|\boldsymbol{\omega}|}{|\boldsymbol{\omega}|^2} M^2 + \frac{\sin|\boldsymbol{\omega}|}{|\boldsymbol{\omega}|} M & \text{if } \boldsymbol{\omega}\cdot\mathbf{m} = 0 \\ \mathrm{I} + M + \frac{1-\cos|\boldsymbol{\omega}|}{|\boldsymbol{\omega}|^2} M^2 + \frac{|\boldsymbol{\omega}|-\sin|\boldsymbol{\omega}|}{|\boldsymbol{\omega}|^3} M^3 & \text{otherwise} \end{cases}$$

It can be shown that for a point $\mathbf{p}$, $(\exp M)\cdot\mathbf{p}$ translates $\mathbf{p}$ if $|\boldsymbol{\omega}| = 0$, rotates $\mathbf{p}$ if $\boldsymbol{\omega}\cdot\mathbf{m} = 0$, otherwise it twists $\mathbf{p}$. The logarithm of a matrix is defined as an inverse of exp.

References

[ACWK04] ANGELIDIS A., CANI M.-P., WYVILL G., KING S.: Swirling-sweepers: Constant-volume modeling. In *Pacific Graphics 2004* (Oct 2004), IEEE.

[Ang04] ANGELIDIS A.: *Hexanions: 6d space for twists*. Tech. Rep. OUCS-2004-20, University of Otago, 2004.

[Bar] BARR A. H.: Global and local deformations of solid primitives. In *Proceedings of SIGGRAPH'84*, pp. 21–30.

[Bat67] BATCHELOR G. K.: *An Introduction to Fluid Dynamics*. Cambridge Univ. Press, UK, 1967.

[CK00] COTTET G.-H., KOUMOUTSAKOS P. D.: *Vortex Methods: Theory and Practice*. Cambridge university press, 2000.

[CMOV02] COTTET G.-H., MICHAUX B., OSSIA S., VANDERLINDEN G.: A comparison of spectral and vortex methods in three-dimensional incompressible flows. *J. Comput. Phys. 175*, 2 (2002), 702–712.

[FL] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. In *Proceedings of SIGGRAPH'04*, pp. 441–448.

[FMa] FOSTER N., METAXAS D.: Modeling the motion of a hot, turbulent gas. In *SIGGRAPH 97 Proceedings*, pp. 181–188.

[FMb] FOSTER N., METAXAS D.: Realistic animation of liquids. In *Graphics Interface '96*, Davis W. A., Bartels R., (Eds.), pp. 204–212.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of SIGGRAPH'01* (Aug 2001), pp. 15–22.

[Gal02] GALLIER J.: More advanced geometric methods in computer science course information. http://www.seas.upenn.edu/~cis70005/, Jan 2002.

[Gha01] GHARAKHANI A.: Grid-free simulation of 3-d vorticity diffusion by a high-order vorticity redistribution method. In *15th AIAA Computational Fluid Dynamics Conference* (Jun 2001), pp. 1–10.

[GLG95] GAMITO M. N., LOPES P. F., GOMES M. R.: Two-dimensional simulation of gaseous phenomena using vortex particles. In *EG Computer Animation and Simulation '95* (Sep 1995), pp. 2–15.

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. In *Proceedings of SIGGRAPH'90* (Aug 1990), pp. 49–57.

[LF02] LAMORLETTE A., FOSTER N.: Structural modeling of natural flames. In *Proceedings of SIGGRAPH 02* (Jul 2002), pp. 729–735.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. In *Proceedings of SIGGRAPH'04* (Aug 2004), vol. 23(3), pp. 457–462.

[LNC91] LIM T., NICKELS T., CHONG M.: A note on the cause of rebound in the head-on collision of a vortex ring with a wall. *Expt. in Fluids 12*, #1/2 (1991), 41–48.

[Mar97] MARGERIT D.: *Mouvement et dynamique des filaments et des anneaux tourbillons de faible épaisseur*. PhD thesis, INPL, 1997.
thesis and papers: http://daniel.margerit.free.fr/pub.html.

[MP89] MILLER G., PEARCE A.: Globular dynamics: A connected particle system for animating viscous fluids. *Computers & Graphics 13*, 3 (1989), 305–309.

[MS98] MCCORMACK J., SHERSTYUK A.: Creating and rendering convolution surfaces. In *Computer Graphics Forum* (Jun 1998), vol. 17(2), pp. 113–120.

[MTPS] MCNAMARA A., TREUILLE A., POPOVIC Z., STAM J.: Fluid control using the adjoint method. In *Proceedings of SIGGRAPH'04*, pp. 449–456.

[Ney03] NEYRET F.: Advected textures. In *Symposium on Computer Animation* (Aug 2003), pp. 147–153.

[PCS04] PIGHIN F., COHEN J. M., SHAH M.: Modeling and editing flows using advected radial basis functions. In *Symposium on Computer Animation* (Aug 2004), pp. 223–232.

[Per85] PERLIN K.: An image synthesizer. In *Proceedings of SIGGRAPH'85* (Jul 1985), pp. 287–296.

[PN01] PERLIN K., NEYRET F.: Flow noise. *Siggraph Technical Sketches and Applications* (Aug 2001), 187.
http://www-evasion.imag.fr/Publications/2001/PN01.

[REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMNER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *Symposium on Computer Animation* (Jul 2004), pp. 193–202.

[RNGF03] RASMUSSEN N., NGUYEN D. Q., GEIGER W., FEDKIW R. P.: Smoke simulation for large-scale phenomena. In *Proceedings of SIGGRAPH'03* (Jul 2003), vol. 22(3), pp. 703–707.

[Rut89] RUTHERFORD A.: *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*, 2nd ed. Dover Publications, Inc, New York, 1989.

[SCP*04] SHAH M., COHEN J. M., PATEL S., LEE P., PIGHIN F.: Extended galilean invariance for adaptive fluid simulation. In *Symposium on Computer Animation* (Jul 2004), pp. 213–221.

[SF93] STAM J., FIUME E.: Turbulent wind fields for gaseous phenomena. In *Proceedings of SIGGRAPH'93* (1993), pp. 369–376.

[SF95] STAM J., FIUME E.: Depiction of fire and other gaseous phenomena using diffusion processes. In *Proceedings of SIGGRAPH'95* (Aug 1995), pp. 129–136.

[SNBW03] SAMAVATI F. F., NUR M. A., BARTELS R., WYVILL B.: Progressive curve representation based on reverse subdivision. In *the 2003 International Conference on Computational Science and Its Applications* (May 2003), pp. 67–78.

[SSEH03] SCHPOK J., SIMONS J., EBERT D. S., HANSEN C.: A real-time cloud modeling, rendering, and animation system. In *Symposium on Computer Animation* (Aug 2003), pp. 160–166.

[Sta99] STAM J.: Stable fluids. In *Proceedings of SIGGRAPH'99* (1999), pp. 121–128.

[Sta01] STAM J.: A simple fluid solver based on the FFT. *Journal of Graphics Tools 6*, 2 (2001), 43–52.

[WH91] WEJCHERT J., HAUMANN D.: Animation aerodynamics. In *Proceedings of SIGGRAPH'91* (Jul 1991).