Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

A Controllable, Fast and Stable Basis for Vortex Based Smoke Simulation

Alexis Angelidis[1] Fabrice Neyret [2] Karan Singh [1] Derek Nowrouzezahrai[1]

[1]Dynamic Graphics Project, U. of Toronto
[2]Laboratoire GRAVIR[†]

Abstract

We introduce a novel method for describing and controlling a 3D smoke simulation. Using harmonic analysis and principal component analysis, we define an underlying description of the fluid flow that is compact and meaningful to non-expert users. The motion of the smoke can be modified with high level tools, such as animated current curves, attractors and tornadoes. Our simulation is controllable, interactive and stable for arbitrarily long periods of time. The simulation's computational cost increases linearly in the number of motion samples and smoke particles. Our adaptive smoke particle representation conveniently incorporates the surface-like characteristics of real smoke.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Computational Geometry and Object Modeling Physically based object modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism Animation

1. Introduction

In the context of a production, an animation or special effects director gives instructions related to the placement of elements in relation to each other on the set, whether virtual or real. Manipulating smoke for this purpose requires a fine level of control that is typically precluded by the chaotic nature of fluids. Traditional smoke simulation tools do not use points and curves for control, even though these handles are popular in animation. Given a workflow for animating smoke similar to existing animation techniques, artists could transfer their skills to choreograph such phenomena. The main objective of our paper is to provide animators with a framework for fast and controllable smoke that automatically maintains its chaotic behavior. Our approach uses control points and curves, and thus it can quickly be integrated into a traditional 3D animation pipeline. We define these controls on top of a physical simulation, with natural transitions between controlled and uncontrolled simulations. Our contributions are:

- A compact basis to simulate a fluid that ensures numerical stability.
- Meaningful controls that allow a user to manipulate this basis.
- A simulation algorithm with a complexity cost that scales linearly in the number of motion samples.

[†] GRAVIR is a joint lab of CNRS, INRIA, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

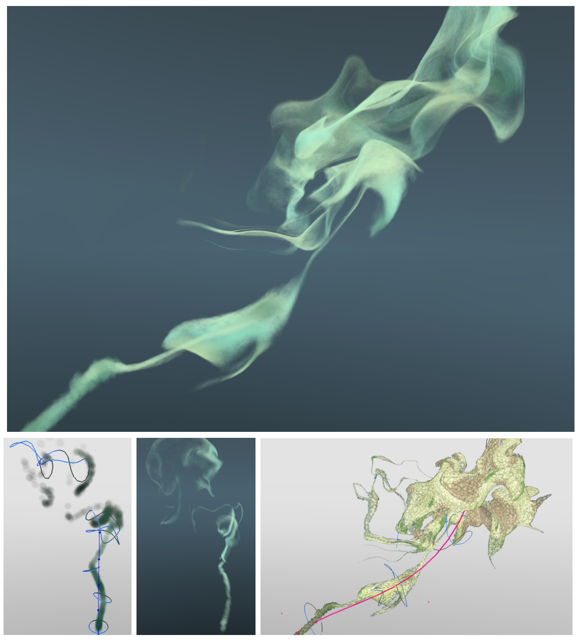
Figure 1: *Bottom left: the motion filaments and plain particles. Bottom middle: corresponding high quality render with splatted ellipsoid particles. Bottom right: the motion filaments with ellipsoid particles' shapes and control curve in red. Top: a controlled fluid simulation.*

© The Eurographics Association 2006.

We will present the novel basis and physical simulator in Section 2. Section 3 will focus on control mechanisms. Smoke particles and rendering will be discussed in Section 4. Algorithms, implementation and hardware acceleration will be discussed in Section 5.

1.1. Related Work

Many approaches have been proposed in Computer Graphics to model gaseous phenomena. In fluid mechanical models, a gas is represented as a quantity carried by a fluid. During the simulation, the large scale motion of the fluid emerges from the definition of small scale update rules and properties modeled as differential equations. These differential equations arise from considering the variation of different quantities defining a flow: velocity [FM, Sta, FSJ], vorticity [YUM, GLG95, PK, AN], circulation [ETK*05], or a combination of them [SRF05]. To solve for the motion, two formalisms may be used: Lagrangian or Eulerian. Solving for the velocity with an Eulerian formalism is a popular approach [FM], which can be made more stable [Sta] and less subject to numerical dissipation [FSJ]. When solving for the velocity using the Eulerian approach, boundary conditions can be handled conveniently on a grid [FM], more adaptively in an octree [LGF04], more accurately near objects with mesh boundaries [FOK05], or with arbitrarily thin objects [GSLF05]. Vorticity methods can also handle boundary conditions to some extent with the panel method [PK].

While simulation techniques aim at producing physically accurate motion, the aim of animation is to control motion in an expressive yet visually plausible manner. Due to the "butterfly effect", small changes in the initial conditions of a fluid produce considerable long term repercussions; thus long term control can hardly be achieved by setting up the initial conditions of a numerical simulation. A structural study of fluid phenomena can yield a trade-off among realism, control and efficiency. This approach has been used successfully for wind [WH] and fire [LF02]. Modest fluid control can be achieved by post-processing a numerical simulation, such as editing the trajectory of particles advected in the flow [PCS]. More computationally intense control can be achieved during the simulation with forces that drive the phenomenon towards an animated target shape [TMPS03, MTPS04, FL04, REN*, SY]: in these methods, any target shape for smoke may be input, but modeling smoke-like shapes is a tedious task. They control the shape envelope of the smoke, while we control the large scale motion trajectory. Recent work achieves control of smoke along a path [KMT].

Our method provides artists with a workflow in which they can animate at a coarse resolution and perform high quality rendering that preserves the motion. Our method is fast and scales linearly in the number of motion samples. Moreover, our method provides users with intuitive controls while maintaining the smoke's chaotic appearance.

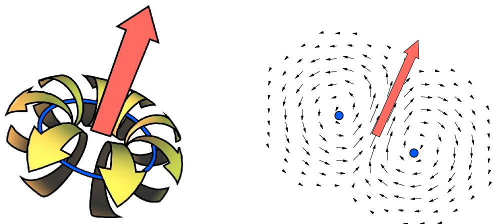

Figure 2: *Left: a loop filament (blue) induces a gust of wind (yellow) traveling in the direction of the loop's axis (red). Right: a two-dimensional slice of the velocity field induced by the filament.*

1.2. Vortex Methods

A fluid may be simulated either by considering the motion of its velocity field v or its vorticity field w. The use of the vorticity field in 3D is a recent development in Computer Graphics [ETK*05, AN, PK, SRF05]. The velocity and vorticity at a point p are related through a pair of mutually inverse relations, the *curl* and the *Biot-Savart law*[†]:

$$\mathrm{w} = \nabla \times \mathrm{v} \tag{1}$$

$$\mathrm{v} = \iiint_{\mathrm{x}} \frac{(\mathrm{p}-\mathrm{x}) \times \mathrm{w}}{4\pi \|\mathrm{p}-\mathrm{x}\|^3}\, \mathrm{dx}\ . \tag{2}$$

Our method is based on the *Lagrangian inviscid vorticity equation*

$$\frac{\mathrm{dw}}{\mathrm{dt}} = \mathrm{w} \cdot \nabla \mathrm{v}\ . \tag{3}$$

Intuitively, the right hand side of this equation means that w is carried by material lines that stretch with the fluid. The motion of a fluid can be solved with Equations (2) and (3): by advecting and stretching a set of markers that carry w (Equation (3)) using the velocity *induced* by the entire field w (Equation (2)). Although computing v directly with the Biot-Savart law requires the vorticity field in the entire space, it is a classic assumption that vorticity concentrates itself in *filaments* [Mar97], and thus a complex flow may be defined compactly with a small set of curves.

The advantages of using the Biot-Savart law with Lagrangian curves carrying w are the intrinsic incompressibility of the flow and the absence of numerical dissipation. Furthermore, vorticity structured in a loop curve can be simply understood: the structure moves along the loop's axis and induces a gust of wind in that direction (see Figure 2). Defining loops is a convenient way to specify initial conditions and manipulate the fluid's motion. Thus our simulation is based on this primitive, although open-curves may also be used in principle. Of the various ways to represent the geometry of a loop, we will propose a scheme appropriate for the simulation and control of smoke.

† The Biot-Savart law gives the simplest inverse of $\nabla\times$. Others are obtained by adding a curless field to the velocity.

2. Basis for Modeling a Fluid

A naïve way to represent a loop filament is to use particles connected with line segments. Issues arise with a simulation that uses this approach: when a filament stretches it becomes undersampled and the deformation of the filament describes an increasingly complex shape. Using high resolution filaments with a level of detail scheme partially solves these issues but still imposes a lifespan on the filaments [AN]. Sampling can also be controlled by splitting unstructured point markers [PK].

We solve the sampling issue by representing a filament as a periodic signal described with a bounded number of harmonic coefficients. At each time step, the filament's geometry is synthesized, advected and analyzed. This three step projection mechanism controls both the filament complexity and the sampling, as well as ensuring stability over *arbitrarily long* periods of time.

2.1. Filament Geometry

The complexity of a signal can be described and controlled in terms of component frequencies. This approach can be applied to the filaments' geometry. Straightforward harmonic analysis of the Cartesian coordinates of the filament's samples would introduce non-negligible parasitic frequencies[‡].

Thus, for each filament we first build a *suitable* coordinate system in which the geometry can be analyzed adequately. Consider a filament of length L defined by the points $c_{i\in[1,n]}$ and their centroid $\tilde{c}$. Let $c(l)$ be the cyclic linear interpolation of the c_i's. The samples define a 3×3 matrix of covariance

$$\frac{1}{n}\sum_i (c_i-\tilde{c})\cdot(c_i-\tilde{c})^{\mathrm{T}}\ . \tag{4}$$

This matrix's eigenvalues $\lambda_x \geq \lambda_y \geq \lambda_z$ correspond to orthogonal unit eigenvectors $e_{i\in\{x,y,z\}}$ which define the plane $(\tilde{c}, e_z)$ that best fits the filament's geometry in the least square sense, as well as defining a local frame $(\tilde{c}, e_x, e_y, e_z)$. We can now define the periodic 3D signal that encodes the geometry of a filament

$$s(\theta) = \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} \cdot (c(L\theta/2\pi) - \tilde{c})\ . \tag{5}$$

The harmonic analysis of each dimension of s [AW95] defines three Fourier series[§]. A new signal s' can be built with a bounded number of Fourier coefficients and the new synthesized curve is obtained with

$$c'(l) = \tilde{c} + \begin{pmatrix} e_x & e_y & e_z \end{pmatrix} \cdot s'(2\pi l/L)\ . \tag{6}$$

Thus each filament is defined by a frame and three Fourier series (see Figure 3), which constitutes our basis. Figure 4 shows four simulations with different bounds on the number of frequencies. We will proceed to explain how filaments induce the velocity field.

‡ A change in the origin or axes of the Cartesian coordinate system may introduce edges to the signal of the geometry.

§ Three arrays of complex coefficients.

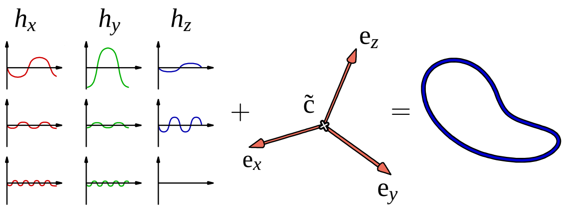

Figure 3: *The description of a filament's geometry (blue) consists of a local coordinate system (red) and three Fourier series (shown on the left as curves).*

2.2. Induced Velocity Field and Dynamics

The velocity field induced by the vorticity is given by the Biot-Savart law whose integral domain reduces to curves thanks to the filament assumption [Mar97]. To describe the vortical structure around a filament, various profile functions can be used. Since the velocity is evaluated at many positions, and since speed is among our objectives, we use a simple piecewise basis function $\tilde{f}$ that does not perform square root computations[¶]. The velocity induced by one filament becomes

$$\bar{v} = \oint_{l\in[0,L]} \gamma\, \tilde{f}(\|p-c\|^2/r^2)(p-c)\times \boldsymbol{\tau}\, dl \tag{7}$$

$$\text{where } \tilde{f}(x^2) = \begin{cases} (4-\frac{20}{x^2+4})^2 & \text{if } x^2<1, \\ 0 & \text{otherwise.} \end{cases}$$

The scalar l parameterizes the filament, $\boldsymbol{\tau}$ is the tangent at c, γ is the *strength* and r is the *thickness*. The above integral can be evaluated discretely at the samples c_i, but using a Riemann sum along the segments (c_i, c_{i+1}) is preferable to avoid motionless spots along the filament when r is small [Ant98]. The velocity v_{total} induced by multiple filaments is the sum of all filaments' velocities. Smoke particles may be advected directly with the above field $\bar{v}$. Advecting and rendering smoke will be the topic of Section 4. In the following section, we explain how filaments are advected.

2.2.1. Filament Dynamics

As the fluid evolves, the Fourier series describing the filament geometries are updated in a three step process:

$$\boxed{\textbf{Geometry Synthesis}} \longrightarrow \boxed{\textbf{Advection}} \longrightarrow \boxed{\textbf{Harmonic Analysis}}$$

Due to the stretching term in Equation (3), the vorticity must change to ensure the preservation of the system's total kinetic energy.

¶ $\tilde{f}$ is monotonic, C^1, numerically very close to C^2, and numerically almost antisymmetric about $1/2$.

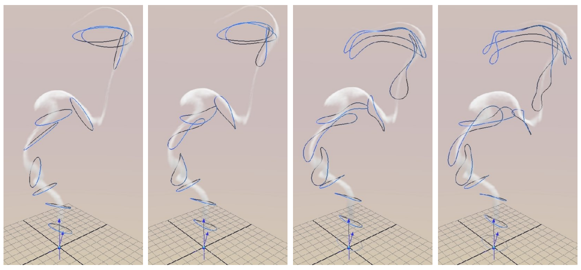

Figure 4: *Simulation of filaments with bounds on the different number of frequencies. From left to right: 1, 2, 4 and 8. Note the similarity in the motion.*

2.2.1.1. Stretching: Kelvin's theorem interprets the stretching term in Equation (3) as the preservation of the circulation when filaments deform [Rut89]. This means that when a filament stretches and its radius decreases, the vorticity must increase, and vice-versa. In practice however, fixing the radius of the filament is important in order to maintain its effect at a meaningful scale and keep a local support. We define the new strength of a filament that stretches from L to L' as $\gamma' = \frac{L}{L'}\gamma$.

3. Animation Controls

Our filament representation holds meaningful information about the fluid. The point $\tilde{c}$ gives an average *location* of the motion, the vector e_z gives an approximate *direction* of the motion and the harmonics define a *complexity texture* over the filament (see Figure 3). This information can be manipulated to adjust the fluid motion.

3.1. Controlling Currents

A naïve attempt to control smoke would be to animate the position of the filaments' frame. This approach does not work since smoke is not directly attached to the filaments and part of the smoke would be left motionless. A more natural manipulation is to modify the filament such that it induces its own motion in the desired direction. This control can be achieved using two operations on filaments: *paddling* and *turning*.

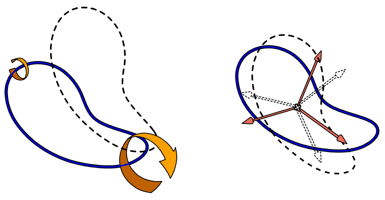

Figure 5: *Controlling filament direction: paddling (left) redistributes the filament's strength, and turning (right) reorients the support of the function.*

3.1.0.2. Paddling: Assuming that the motion of a filament is mostly self-induced, filament *paddling* consists of locally redistributing the filament strength such that the filament induces a gust of wind in the desired direction. To make a filament take a turn around a unit vector m, we define the redistribution function along the filament as $\phi(l) = \mathrm{m} \cdot \boldsymbol{\tau} \in [0,1]$. This function is conservative, since $\oint_l \mathrm{m} \cdot \boldsymbol{\tau}\, dl = 0$. The multiplication of the strength by $1 + k_p\phi$, where k_p is the *amount of paddling*, strengthens the vortices pointing in the direction of m and weakens all other vortices, thus steering the filament in the desired direction.

3.1.0.3. Turning: Since paddling is defined locally, it may be required to enforce the actual direction taken by a filament due to the influence of all the filaments. Since e_z gives the direction taken by a filament, the filament's frame can be rotated to align e_z with a desired direction. Paddling is more preferable than turning since turning slightly displaces the support of the motion.

In practice, paddling is set to a default value, while the amount of turning is a user controlled parameter. The two operations are shown in Figure 5.

3.1.1. Current Curve

From a user standpoint, controlling a flow by defining a *current* is an intuitive concept. Smoke is often depicted depicted with curves in traditional animation (see Figure 6). Our simulator can achieve this by driving the filament with the tangents of a curve. One possible way to assign a tangent of a curve to a filament is to find the closest point on the curve at every step. The closest point is not uniquely defined and is computationally expensive. Alternatively, we propose the use of a local method.

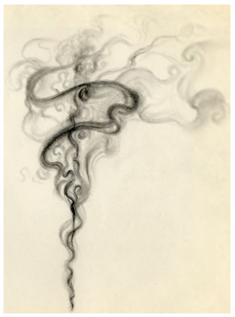

Figure 6: *Fantasia (1940), Disney©.*

Given a control curve K, an arc-length parameter d along the curve is assigned to the filament. During a time step, the filament is first updated normally without applying any constraints. This provides its previous and next centroids, $\tilde{c}$ and $\tilde{c}_{\text{next}}$. The filament's new arc-length parameter is $d' = d + \|\tilde{c} - \tilde{c}_{\text{next}}\|$. The new direction is given by the tangent of the curve $e'_z = \boldsymbol{\tau}(d')$. The constrained centroid $\tilde{c}'$ is a projection of $\tilde{c}_{\text{next}}$ on the plane $(K(d'), \boldsymbol{\tau}(d'))$. Note that e_x and e_y must be updated as well. This constraint mechanism makes computation stable, local and inexpensive. The method is illustrated in Figure 7.

3.1.2. Current Attractor

An attractor is another intuitive tool that can be applied to our fluid description. A user can control the flow by placing

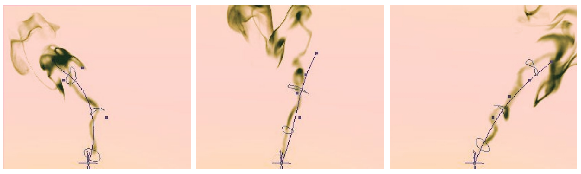
Figure 7: *To control the motion, the user animates a curve.*

a point that will make the filaments paddle towards it. Once a filament has passed through the attractor, it is marked and released, to prevent filaments from all clustering in one area.

3.1.3. Tornado-like Effect

Twisting of the smoke along the motion direction can be added to achieve tornado-like effects. This can be done by modifying the spinning direction of the vortices according to the local coordinate frame of each filament. The following is the twisting velocity

$$\mathrm{v}_{\text{tornado}} = \sum_{i=1}^{n} \tilde{f}(\|\mathrm{p}-\mathrm{c}_i\|^2/r^2)(\mathrm{p}-\mathrm{c}_i)\times \mathrm{e}_z \qquad (8)$$

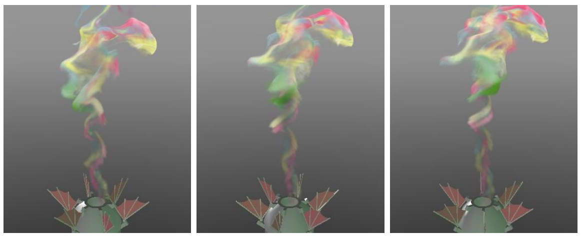
Figure 8: *The smoke twists along its motion in a controlled manner by adding an external velocity.*

3.2. Noise

The addition of visual detail over a simulation has been addressed in Computer Graphics [SF, RNGF03]. We propose a technique to allow users to place zones of turbulence and add convincing detail to the flow in a desired area. Our technique is local, divergence free, computationally inexpensive and uses a very low amount of memory.

Assuming that small scale perturbations do not affect the controllable large scale motion of the filaments, small scale motion detail can be added convincingly over the existing motion. The detail is generated by performing a simple simulation in a toroidal cube space using vortex particles and ignoring particle stretching. For the noise to be convincing, the radius of the noise vortices must be small enough compared to the thickness of the filaments[‖]. Since the noise simulation is tileable, it can be replicated to cover a larger area. The area can be placed seamlessly in the scene by using a falloff function across the area's boundary. To preserve the fluid's incompressibility, the falloff function modulates the strength

[‖] Half or less of the thickness of the filaments.

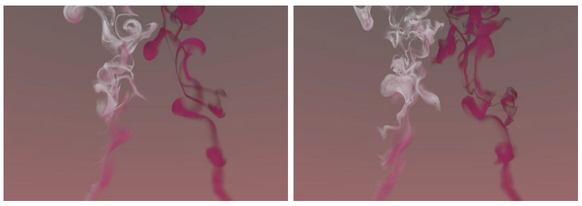
Figure 9: *Noise disabled (left) and enabled (right). The large scale motion remains identical.*

of the noise vortices near the boundary as opposed to modulating the velocity. Because motion induced by the noise is added to the motion induced by the filaments, and because the motion of the fluid is only observable where there are smoke particles, we have never observed the repetition of a smoke pattern.

4. Smoke Advection and Rendering

One advantage of simulating the fluid motion with a Lagrangian approach is the unbounded area of simulation. Simulating smoke with particles as opposed to a grid retains this advantage. For the task of animating and rendering *thick smoke*, plain particles are suitable. For *thin smoke*, it is preferable to use adaptive particles and accumulate each particle's stretching. We briefly summarize the deformation of a particle below [AN].

At each time step, the new position of a particle is given by $\mathrm{x}_{i+1} = \mathrm{x}_i + \delta t\ \mathrm{v}_{\text{total}}(\mathrm{x}_i, t_i)$ or a higher order reconstruction (see Appendix A). The stretching during one time step is measured at a point with the *displacement gradient tensor* of that step: $\frac{\mathrm{dx}_{i+1}}{\mathrm{dx}_i}$, where x_i is the position at time t_i. The stretching accumulated during a longer period of time is given by the chain rule

$$\frac{\mathrm{dx}_n}{\mathrm{dx}_0} = \frac{\mathrm{dx}_n}{\mathrm{dx}_{n-1}} \cdot \frac{\mathrm{dx}_{n-1}}{\mathrm{dx}_{n-2}} \cdots \frac{\mathrm{dx}_1}{\mathrm{dx}_0}, \qquad (9)$$

where n is the number of time steps. The ellipsoidal shape of the particle is defined by the square roots of the eigenvalues and the eigenvectors of the *metric tensor* $\frac{\mathrm{dx}_n}{\mathrm{dx}_0} \cdot \frac{\mathrm{dx}_n}{\mathrm{dx}_0}^{\mathrm{T}}$.

4.1. Shading

Self-shadowing is an important visual cue to convey the 3-dimensionality of smoke. We use a fast self-shadowing algorithm [BMC05]. We also notice that real thin smoke displays surface-like properties, as shown in Figure 12. We propose to simulate this behavior to add realism. We define the normal to the smoke as the gradient of the smoke density. For a de-

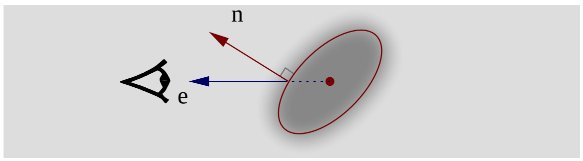

Figure 10: *The vector* n *indicates a local gradient of density.*

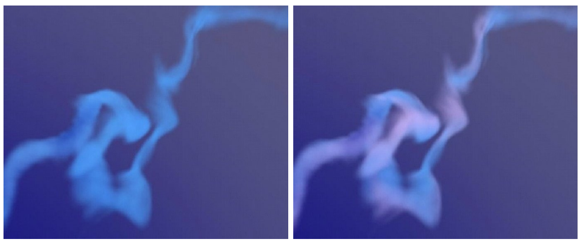

Figure 11: *Diffuse reflection disabled (left) and enabled (right).*

formable ellipsoid particle, normals to the ellipsoid correspond to gradients of density near the particle's boundary. Computing this normal when the view vector passes through the ellipsoid's center as shown in Figure 10 is straightforward

$$\mathrm{n} = \left(\frac{\mathrm{dx}_n}{\mathrm{dx}_0} \cdot \frac{\mathrm{dx}_n}{\mathrm{dx}_0}^{\mathrm{T}} \right)^{-1} \cdot \mathrm{e} \,. \tag{10}$$

The normal n can be used in a local illumination shading model, such as the Blinn-Phong model [Bli].

5. Algorithm and Implementation

We summarize our method of simulation and provide the necessary implementation details. One time step of the simulation can be decomposed into five main steps:

1. Synthesize all the filament geometries using their Fourier series and local coordinates by connecting points on the filament computed with Equation (6).
2. Advect the smoke particles using the velocity field induced by the filament geometries using Equation (7). If the particles are deformable, compute particle stretching (see Section 4).
3. Advect each filaments' geometry using the velocity field induced by the filament geometries, and modulate their strength to account for stretching (see Section 2.2.1).
4. Compute each filament's new local coordinates system with principle component analysis, and perform the harmonics analysis of the filaments to control sampling and complexity (see Equation (5)).
5. Apply the controls to each filament's representation (see Section 3).

5.1. Complexity

To make this algorithm scalable for a large number of filaments, special care must be taken in the third step. Although filaments have a local influence of size r, the advection of N filaments with Equation (7) would have complexity $O(N^2)$. This complexity can be reduced by caching the velocity in a dynamic octree subdivided only near the smoke and the filaments, and interpolating the velocity between leaves. With this caching mechanism, our method is *linear* in the number of filaments in the worst case. The motion can be defined with a higher accuracy by storing *screw transforms* in the octree instead of velocities (see Appendix A). Advection can be accelerated using commodity graphics hardware.

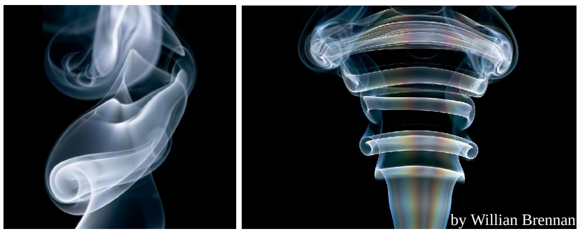

Figure 12: *Photographs of incense smoke show the reflective (left) and transmissive (right) character of incense smoke.*

5.2. Hardware Acceleration

Recent advances in the field of general programming on graphics processing units (GPUs) have shown that it is possible to accelerate physical simulations on GPUs [KW05, KLRS04]. In our simulation, the advection of particles can benefit from hardware acceleration. In Appendix A, we describe a higher order trajectory reconstruction method. While it is more accurate than advecting with the velocity of Equation (7), it is also computationally more expensive. We propose a GPU implementation to accelerate this process. Although the latest generation of GPUs are capable of achieving remarkable computational throughput, the memory bus bandwidth limitations introduce performance bottlenecks that must be taken into account.

At every step of the simulation, the particle positions and their associated screw transforms are packed into texture maps and transferred to the GPU for processing. The exponentiation of the screw transform and position updates are implemented as shader operations. Once the operations are complete, the updated positions are returned to the CPU for use in the next frame. Using the OpenGL 2.0 pixel buffer object and floating point render target extensions yielded the optimal data transfer rates and minimal per-frame setup overhead. The performance gain of the hardware accelerated implementation becomes more prevalent as the number of particles increases.

6. Performance

The performance of our method is measured in Table 1 using examples created with our toolkit. Each scene's performance is measured twice with two different user settings: *modeling* and *high-quality*. In the high-quality settings, parameters are set to produce the final rendering of an animation. This includes effects such as noise, adaptive particle splitting and a high number of component frequencies. In the modeling settings, the number of filament samples and the number of sampling harmonics are reduced while still preserving seemingly identical motion to the high-quality settings results. Modeling settings reproduce the settings an artist uses when creating a scene.

The scene complexity is measured with the average number of filament particles per frame, the average number of smoke particles per frame and whether the scene takes advantage of particle splitting and noise effects. The perfor-

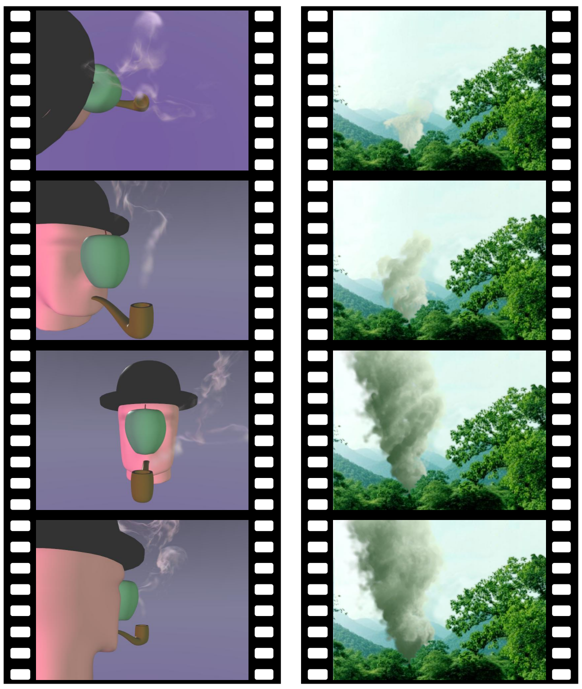

Figure 13: *Left: example of a controlled smoke animation. As the camera moves, it is easy for the animator to ensure that the smoke is always visible and does not cover the face. Right: example of a uncontrolled smoke animation. Although many of our examples show thin smoke, our method is also suitable for thick smoke.*

mance is measured in *average* frame rates over a 200 frames animation. The results in Table 1 illustrate the linear scalability of our method in the number of particles and filaments. Interactive framerates are achieved in all modeling settings simulations, and some less complex high-quality settings simulations.

7. Conclusion

We have proposed a compact and meaningful basis for fluids that captures complex flows. Using this basis, we have illustrated how to manipulate the fluid's motion with general tools such as current curves, attractors and tornado-like effects. Transitions between a controlled and uncontrolled numerical simulation are achieved seamlessly. Depending on the visual requirements of a shot, more specific modifications may be applied to the fluid, such as making the filaments follow the gradient of a scalar field. Our physical simulation is scalable, thus providing a user-friendly workflow: low-resolution editing with few particles and high-resolution rendering that preserves the original motion.

Appendix A: Higher Order Trajectory

Geometrically, the Biot-Savart law expresses the velocity induced by w at p as the velocity of simultaneous rotations of angle $\frac{1}{4\pi\|p-x\|^3}$ (see Section 1.2). The resulting velocity is tangent to the trajectory of the particle, and can be used to approximate the trajectory by a translation. However, the geometric transformation that results from applying simultaneous rotations is a *screw*, i.e. a transformation with a translation and a rotation component [Gal00]. The Biot-Savart law can be rearranged to make the screw visible in the form of a 4×4 matrix that multiplies p, by expressing the cross product as a matrix

$$v = \left[\iiint_x \frac{-1}{4\pi\|p-x\|^3} \begin{pmatrix} w\times & & & x \times w \\ 0 & 0 & 0 & 0 \end{pmatrix} dx \right] \cdot p \,.$$

If we call M the above matrix, it belongs to the Lie algebra $\mathfrak{se}(3)$, and defines the trajectory of p as $\exp(t\ M) \cdot p$, and the new position after a time step of length δt is $\exp(\delta t\ M) \cdot p$ [Gal00], where exp is defined as $\sum_{k=0}^{\infty} \frac{M^k}{k!}$. In the case of one filament with a discrete integral and our fast vortex distribution function, M becomes

$$M = \sum_{i=0}^{n} -\gamma \tilde{f}(\|p-c_i\|^2/r^2) \begin{pmatrix} \tau_i\times & & & c_i \times \tau_i \\ 0 & 0 & 0 & 0 \end{pmatrix} ,$$

which provides a more accurate trajectory than (7).

References

[AN] Angelidis A., Neyret F.: Simulation of Smoke Based on Vortex Filament Primitives. In *SCA'05: Proc. of the Symposium on Computer Animation*, pp. 87–96.

[Ant98] Anton H.: *Calculus: A New Horizon*. John Wiley & Sons, 1998.

[AW95] Arfken G., Weber H.: *Mathematical Methods for Physicists*. Academic Press, 1995.

[Bli] Blinn J.: Models of Light Reflection for Computer Synthesized Pictures. In *ACM Trans. Graph. (Proc of SIGGRAPH'77)*, pp. 192–198.

[BMC05] Bertails F., Ménier C., Cani M.-P.: A Practical Self-Shadowing Algorithm for Interactive Hair Animation. In *GI* (2005), pp. 71–78.

[ETK*05] Elcott S., Tong Y., Kanso E., Shröder P., Desbrun M.: Stable, Circulation-Preserving, Simplicial Fluids. In *Discrete Differential Geometry*, Chapter 9 of course notes. ACM SIGGRAPH, 2005.

[FL04] Fattal R., Lischinski D.: Target-Driven Smoke Animation. *ACM Trans. Graph. 23*, 3 (2004), 441–448.

[FM] Foster N., Metaxas D.: Modeling the Motion of Hot, Turbulent Gas. In *ACM Trans. Graph. (Proc of SIGGRAPH'97)*, pp. 181–188.

[FOK05] Feldman B. E., O'Brien J. F., Klingner B. M.: Animating Gases with Hybrid Meshes. *ACM Trans. Graph. 24*, 3 (2005), 904–909.

[FSJ] Fedkiw R., Stam J., Jensen H. W.: Visual Simulation of Smoke. In *ACM Trans. Graph. (Proc of SIGGRAPH'01)*, pp. 15–22.

[Gal00] Gallier J.: *Geometric Methods and Applications: For Computer Science and Engineering*. Springer, 2000.

Scene	Modeling Settings			High-Quality Settings				
	Avg. # of Motion Samples	Avg. # of Particles	Avg. # of FPS	Particle Splitting	Noise	Avg. # of Motion Samples	Avg. # of Particles	Avg. # of FPS
Happy Boy	113	72	21.71	✓		450	1197	3.61
Simple Benchmark	78	96	18.43			90	168	7.63
Toxic Fumes vs. the Fly	45	112	18.32	✓	✓	182	3787	0.37
Noise Benchmark	91	112	12.63	✓	✓	364	3038	2.516
Genie & Lamp	67	2493	12.435	✓	✓	450	73357	0.24
Forest Fire	120	4875	5.16		✓	336	20250	0.54
Walkthrough Smoke	395	1280	4.744	✓	✓	750	7040	1.03

Table 1: *Benchmarks with modeling settings and high-quality settings, using a Pentium 4 2.4GHz with 256MB of RAM. The performance values listed here were not generated with the GPU accelerated advection feature (see Section 5.2). This eliminates the bias introduced by GPU performance enhancement.*

[GLG95] Gamito M., Lopes P., Gomes M.: Two-Dimensional Simulation of Gaseous Phenomena using Vortex Particles. In *EG Computer Animation and Simulation '95* (1995), pp. 2–15.

[GSLF05] Guendelman E., Selle A., Losasso F., Fedkiw R.: Coupling Water and Smoke to Thin Deformable and Rigid Shells. *ACM Trans. Graph. 24*, 3 (2005), 973–981.

[KLRS04] Kolb A., Latta L., Rezk-Salama C.: Hardware-Based Simulation and Collision Detection for Large Particle Systems. In *HWWS'04: Proc of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2004), pp. 123–131.

[KMT] Kim Y., Machiraju R., Thompson D.: Path-Based Control of Smoke Simulations. In *SCA'06: Proc. of the Symposium on Computer Animation*.

[KW05] Krüger J., Westermann R.: GPU Simulation and Rendering of Volumetric Effects for Computer Games and Virtual Environments. *Computer Graphics Forum 24,3* (2005).

[LF02] Lamorlette A., Foster N.: Structural Modeling of Flames for a Production Environment. *ACM Trans. Graph. 21*, 3 (2002), 729–735.

[LGF04] Losasso F., Gibou F., Fedkiw R.: Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph. 23*, 3 (2004), 457–462.

[Mar97] Margerit D.: *Mouvement et Dynamique des Filaments et des Anneaux Tourbillons de Faible Epaisseur*. PhD thesis, INPL, 1997.

[MTPS04] McNamara A., Treuille A., Popović Z., Stam J.: Fluid Control Using the Adjoint Method. *ACM Trans. Graph. 23*, 3 (2004), 449–456.

[PCS] Pighin F., Cohen J. M., Shah M.: Modeling and Editing Flows Using Advected Radial Basis Functions. In *SCA'04: Proc. of the Symposium on Computer Animation*, pp. 223–232.

[PK] Park S., Kim M.: Vortex Fluid for Gaseous Phenomena. In *SCA'05: Proc. of the Symposium on Computer Animation*, pp. 261–270.

[REN*] Rasmussen N., Enright D., Nguyen D., Marino S., Sumner N., Geiger W., Hoon S., Fedkiw R.: Directable Photorealistic Liquids. In *SCA'04: Proc. of the Symposium on Computer Animation*, pp. 193–202.

[RNGF03] Rasmussen N., Nguyen D. Q., Geiger W., Fedkiw R.: Smoke Simulation for Large Scale Phenomena. *ACM Trans. Graph. 22*, 3 (2003), 703–707.

[Rut89] Rutherford A.: *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Dover Publications, Inc, 1989.

[SF] Stam J., Fiume E.: Turbulent Wind Fields for Gaseous Phenomena. In *ACM Trans. Graph. (Proc of SIGGRAPH'93)*, pp. 369–376.

[SRF05] Selle A., Rasmussen N., Fedkiw R.: A Vortex Particle Method for Smoke, Water and Explosions. *ACM Trans. Graph. 24*, 3 (2005), 910–914.

[Sta] Stam J.: Stable Fluids. In *ACM Trans. Graph. (Proc of SIGGRAPH'99)*, pp. 121–128.

[SY] Shi L., Yu Y.: Taming Liquids for Rapidly Changing Targets. In *SCA'05: Proc. of the Symposium on Computer Animation*, pp. 229–236.

[TMPS03] Treuille A., McNamara A., Popović Z., Stam J.: Keyframe Control of Smoke Simulations. *ACM Trans. Graph. 22*, 3 (2003), 716–723.

[WH] Wejchert J., Haumann D.: Animation Aerodynamics. In *ACM Trans. Graph. (Proc of SIGGRAPH'91)*, pp. 19–22.

[YUM] Yaeger L., Upson C., Myers R.: Combining Physical and Visual Simulation-Creation of the Planet Jupiter for the Film "2010". In *ACM Trans. Graph. (Proc of SIGGRAPH'86)*, pp. 85–93.

© The Eurographics Association 2006.