Real-time Interactive Simulation of Smoke Using Discrete Integrable Vortex Filaments.

Conference Paper · January 2009

DOI: 10.2312/PE/vriphys/vriphys09/001-010 · Source: DBLP

2 authors:

Steffen Weissmann
Technische Universität Berlin
15 PUBLICATIONS **125** CITATIONS
SEE PROFILE

Ulrich Pinkall
Technische Universität Berlin
127 PUBLICATIONS **4,068** CITATIONS
SEE PROFILE

Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2009)
H. Prautzsch, A. Schmitt, J. Bender, M. Teschner (Editors)

Real-time interactive simulation of smoke using discrete integrable vortex filaments

Steffen Weißmann and Ulrich Pinkall

Institut für Mathematik, Technische Universität Berlin, Germany

Abstract

We present a fluid solver for the real-time interactive simulation of inviscid, ideal fluid flow. The simulation is based on the evolution of discrete vortex filaments, which allow a dramatic increase of detail and performance compared to traditional methods used in Computer Graphics. As a fully lagrangian method the simulation is not restricted to a fixed domain and does not suffer from numerical dissipation. Vortex filaments arise naturally in real flows and thus provide an excellent building block for modelling realistic smoke. We present a GPU-based implementation which allows the interactive experimentation with 3D fluid flow on desktop computers and also in distributed immersive virtual environments.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation; Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

Real-time simulation of 3D smoke is an important ingredient for virtual environments in general and computer games in particular. Nevertheless computational challanges have so far prevented the widespread implementation of such simulations. While it is possible to achieve realistic and highly resolved 3D smoke animations in extensive offline simulations, real-time applications are still missing the desireable detail and realism.

We describe a 3D method that is highly efficient while allowing a tremendous amount of detail. The method is based on the simulation of the evolution of vortex filaments. Vorticity originates as 2–dimensional vortex sheets that tend to roll up into complicated 1–dimensional structures. The resulting visual complexity easily exceeds the level of detail that can be achieved with real-time grid based methods, as in Figure 1. The use of vortex filaments provides an efficient method to capture the complexity of smoke with very sparse data. The whole fluid velocity field is defined by the vortex filaments and can be used to advect arbitrary marker particles. Besides the application for real-time simulation, the method provides a significant improvement for the workflow of effects artists designing smoke animations. It allows to obtain an immediate preview of the fluid motion. The final animation can then be rendered with an arbitrary number of particles – without affecting the fluid motion at all.

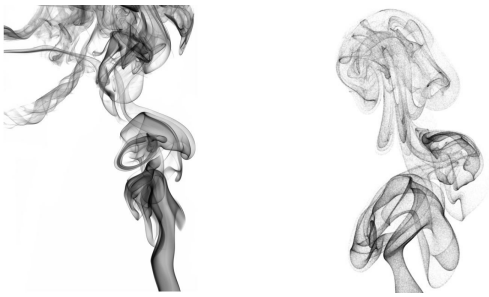

Figure 1: *Comparison of a photograph (left, from [Jef]) with our simulation (right). The 396^2 particles are rendered as unshaded transparent GL points.*

2. Related work and contribution

Real-time 3D smoke simulations have so far been restricted to low resolution grid-based Eulerian methods (mostly based on Stam's *Stable Fluids* [Sta99] with vorticity confinement [FSJ01]) or to algorithms based on 2D reductions

[KW05]. Recent progress with GPU techniques allows relatively large 3D grids [LLW04, CLT07, Yan09a, Yan09b] for such methods. Model reduction [TLP06] can significantly speed up such simulations, but it requires extensive precomputations as well as giant storage, which limits its practical applicability to very small grids. Also hybrid methods based on [SRF05] were applied to real-time applications, for instance in [EVG07].

Smoothed Particle Hydrodynamics methods [SF95, DG96, MCG03, BT07] can be used at interactive rates with a respectable number of particles thanks to GPU techniques [ZSP07, YWH*09]. These methods are used with great success for the simulation of water and other liquids. They are however not well-suited for inviscid fluids such as smoke.

The use of vortex filaments as basic primitives for modelling 3D smoke was pioneered by Angelidis and Neyret [AN05, ANSN06]. They however simulated the motion of smoke using a kernel function that differs significantly from the physical system, resulting in an incorrect asymptotic falloff of the velocity field.

Our method is based on the direct discretization of the actual physics. The fluid motion is described by a Hamiltionian system, which guarantees the conservation of both energy and momentum. We are able to explicitly integrate the correct kernel function for the discretized vortex filaments. This yields not only an important increase of physical realism but at the same time in fact a reduction of the computational costs. Our second contribution is the following: When discretizing the vortex filaments as polygons, their velocity will be significantly underestimated when calculated by the integral kernel alone: Both adjacent edges do not contribute at all to the velocity of a vertex. This effectively results in modelling vortex filaments that have a thickness equal to the edge lengths. This means that excessively many edges would be required to model thin (and therefore fast) filaments. We compensate this discretization artefact by incorporating a discrete evolution equation for polygons (developed in [PSW07]), that excellently captures the locally induced speed. Also this local contribution to the evolution conserves energy and momentum and can be computed efficiently. We are therefore able to model thin vortex filaments using only a small number of edges.

We will describe an interactive, GPU accelerated implementation for use in distributed, immersive virtual environments and on desktop computers.

3. Physics of vortex filaments

Irrotational regions will usually be rare given an arbitrary divergence free velocity field. The fluids that we are going to simulate (represented by a small number of vortex filaments) are mostly irrotational, except for thin tubular regions around the filaments. Therefore some explanation is required why it is nevertheless a good idea to model realistic fluids in this way: First, in inviscid fluid, there is no way to increase the volume that contains vorticity. Second, 2–dimensional vortex sheets tend to roll up into 1–dimensional core structures [RPN00] (see Figure 2). This implies that vortex filaments will actually stay filaments, since any flattening of the core structure will result in a rollup. Most important, vorticity usually originates as vortex sheets, for instance at boundary layers or around jets. Thus we can excellently model realistic fluids, using a small number of vortex filaments.

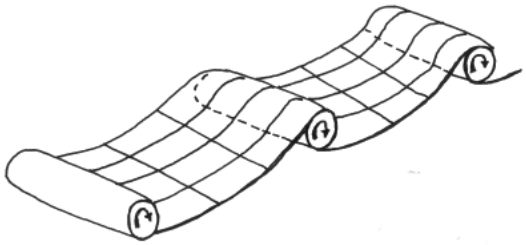

Figure 2: *Sketch of the rollup of a vortex sheet to vortex filaments, from [RPN00].*

Besides the physical motivation, vortex filaments are a very convenient model: They move along with the motion of the fluid, they do not split, join or vanish. They do not change their strength, so only the geometry of the filaments needs to be tracked. From an artist's point of view, filaments give an intuitive primitive to model fluid motion. Another very important property of vortex filaments is that they ensure that the *vorticity field* is always divergence free. In contrast, vortex particle methods [PK05] only guarantee a divergence free velocity field – the vorticity field will not stay divergence-free when the vortex structures break up.

3.1. Mathematical description

The motion of an inviscid, ideal fluid is described by a time-dependent divergence free velocity field u, obeying *Euler's equation*. The vorticity field $\omega = \operatorname{curl} u$ is again a divergence free vector field, and we assume that it is compactly supported. The vorticity field lines are either closed or end on boundaries, which is not a strict mathematical consequence but based on the observation how vorticity is created in real fluid flow. Euler's equation in vorticity form

$$\dot{\omega} = \operatorname{curl}(\omega \times u) \tag{1}$$

states that the vorticity field is forward advected by the velocity field just as the fluid material. The right hand side of Equation (1) is the *Lie bracket* $[\omega, u]$ of divergence free vectorfields in $\mathbb{R}^3$.

From the vorticity field ω we can reconstruct the velocity

field u using the formula

$$u(x) = -\frac{1}{4\pi}\int_{\mathbb{R}^3} \frac{x-z}{\|x-z\|^3} \times \omega(z)\,dz. \tag{2}$$

This is an integral over the whole of $\mathbb{R}^3$, but with the restriction to vorticity fields that are supported on tubular neighbourhoods of closed space curves γ_k, Equation (2) reduces to a sum of line integrals – the *Biot-Savart law*:

$$u(x) = \sum_k -\frac{\Gamma_k}{4\pi}\oint \frac{x-\gamma_k(s)}{\|x-\gamma_k(s)\|^3} \times \gamma_k'(s)\,ds, \tag{3}$$

Here, Γ_k is the *strength* of the filament, it is the flux of vorticity across any cross section of the filament core. The strength is constant along the filament and also constant in time, by Helmholtz's theorems and also as a direct consequence of Equation (1).

For the following discussion we will regard one single vortex filament γ, the results can be transfered to a set of filaments by superposition (i.e. summation) of the individual vector fields.

The dynamical system we want to simulate is

$$\dot\gamma(s) = u(\gamma(s)) \tag{4}$$

where u is given by Equation (3). The problem is that the velocity field u diverges logarithmically for points on the γ. Assuming a small but finite tube radius of the vortex filament γ reveals that the filament moves according to the *smoke ring flow*

$$\dot\gamma(s) = K\gamma'(s) \times \gamma''(s), \tag{5}$$

where the constant K is proportional to the strength Γ of γ. The smoke ring equation (5) is obtained by a renormalization from Equation (4), thus it conserves both energy and momentum of the original system. It was discovered at the beginning of the 20th century by Da Rios [SDR06], its integrability was found by Hasimoto [Has72]. A historical review is given in [Ric91].

The correct speed of a smooth vortex filament with small thickness a is obtained by applying a *cut off* to the right-hand-side of Equation (4): A small portion around the evaluation point is excluded from the integration domain. This is known [Saf92] to be equivalent to replacing the singular Bio-Savart kernel by the *Rosenhead-Moore* kernel, which gives the velocity field

$$u(x) = -\frac{\Gamma_i}{4\pi}\oint \frac{x-\gamma(s)}{\sqrt{a^2+\|x-\gamma(s)\|^2}^3} \times \gamma'(s)\,ds, \tag{6}$$

where a (the thickness parameter) depends on the core radius of the filament. Another way to obtain Equation (6) is to apply a fixed convolution kernel to the singular vorticity field which is concentrated along the filament (see [PSW07]). For $a > 0$ this velocity field has no singularities at all, which makes its evaluation unproblematic.

The singular Biot-Savart formula (3) causes two problems to the numerics: The singular behavior at the filaments and the fact that it is hard to integrate analytically – already for circular filaments it results in elliptic functions. Angelidis and Neyret [AN05] have addressed these issues by a modification of the kernel function. To ease analytical integration, they have changed the exponent in the denominator from 3 to 4. The resulting equation of motion is no longer related to the physical system, for instance the asymptotic falloff is different and the system does not scale correctly.

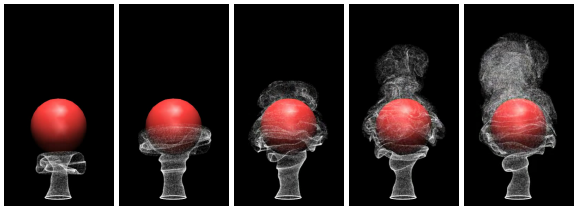

Figure 3: *Flow around a sphere using image vorticity.*

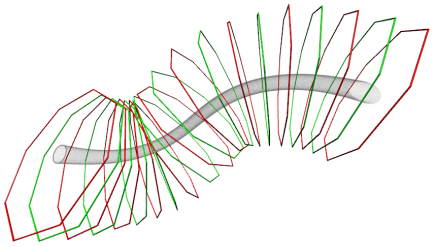

Figure 4: *Desinging irrotational background velocity fields using static vortex filaments. The particles were injected from a circular curve at the left and advected by the induced velocity field from the static filaments.*

3.2. Boundary conditions and background flow

We will not deal with boundary conditions here. Slippy boundaries can be modelled using *image vorticity* [Saf92], for instance an infinite ground plane (as in [AN05]) is obtained by mirroring all filaments at the plane. A sphere can be modelled by inversion of the vortex filaments at the sphere (see Figure 3). The resulting velocity field will be tangent to the surface, but in contrast to real boundaries, no vorticity is created. We will not go into details about these special cases for dealing with boundaries. We will describe the incorporation of arbitrary obstacles in future work.

As a background flow, any irrotational field can be added to the velocity field given by Equation (6). In this way simple effects of wind or temparature lift can be simulated. It is however not possible to combine this with the image vorticity construction described above, except for the case that the background flow is already tangent to the boundaries. On the other hand it is difficult to design an irrotational background field with certain features. Therefore we suggest the use of

static vortex filaments, that contribute to the velocity field but do not get advected by the flow: Such filaments generate a velocity field that is irrotational except for some small region around the filament. In this way one can easily design a background field that for instance guides the fluid motion along a certain path, see Figure 4. This gives artists a very intuitive tool to control large scale features of the fluid motion. In addition, such background flows can be combined with the simple image vorticity constuctions.

3.3. Conservation laws

The smooth system we have described is a *Hamiltonian system* (see [EM70, AK98]), preserving the energy

$$E = \sum_{i,j} \frac{\Gamma_i \Gamma_j}{8\pi} \oint \oint \frac{\langle \gamma_i(s), \gamma_j(\tilde{s}) \rangle}{\sqrt{a^2 + \|\gamma_i(s) - \gamma_j(\tilde{s})\|^2}} \, ds \, d\tilde{s}. \tag{7}$$

This energy has a nice geometric interpretation: It is the sum of pair-wise fluxes, that the filaments induce to each others. Physically, the energy is very closely related to the kinetic energy $T = \int_{\mathbb{R}^3} \langle u(x), u(x) \rangle dx$, in fact it is the kinetic energy of a fluid with the same vortex filaments, but a slightly different core structure and radius. Another constant of motion is the *hydrodynamic impulse*

$$A = \sum_i \frac{1}{2} \oint \gamma_i(s) \times \gamma_i'(s) \, ds \tag{8}$$

whose geometric interpretation (for closed filaments) is the following: For any plane $n^\perp$ with unit normal vector n, the signed area of the orthogonal projection of the filaments onto that plane is $\langle A, n \rangle$. As a reference for Equations (7) and (8) see [Saf92].

This invariant has an important consequence for the geometry of the filaments (see [Cho91]): Vortex filaments stretch rapidly while evolving under the fluid flow. At the same time, their projected area (for instance onto $A^\perp$) stays constant. This is only possible when the filaments fold, leading to very long and highly curved filaments. While folding, nearby sections of the vortex filaments tend to align anti-parallel, resulting in effectively canceling out each others' contribution to the overall velocity field. A special case is the creation of *hairpins* [Cho90, Cho93]. In slightly viscous fluids, vortex reconnections occur at such anti-parallel aligned filaments.

4. Polygonal discretization of smooth filaments

We start with the smooth hamiltonian system described in Section 3.1 and discretize it by replacing the smooth vortex filaments by polygons and advecting the polygon vertices according to the dynamical system (4). In Section 4.1 we will describe in detail the explicit integration of the velocity field (Equation (6)) that is generated by a straight filament edge. The resulting velocity field is obtained by summing up over all edges. For a single step of the simulation we will integrate the ODE (4) over the current time step Δt using a standard forward integration scheme.

It turns out that, due to the discretization as polygons, the locally induced velocity of the filaments gets lost. This happens since adjacent edges do not contribute to the velocity at a vertex. The actual locally induced speed is proportional to the smoke ring flow (5), which we cannot compute directly for polygons. Instead, we apply one step of a time-discrete evolution equation (the doubly-discrete smoke ring flow for polygons derived in [PSW07]) that acts as a time integrator of the smoke ring flow for polygons. In Section 4.3 we will explain how to compute a single step of the evolution and how to determine the necessary parameters.

4.1. Explicit integration of the Biot-Savart law

If γ is a piecewise linear parametrisation of a closed polygon, on each edge we have $\gamma'' = 0$ and we find an explicit anti-derivative for the integrands of equation (6):

$$\frac{\langle \gamma, \gamma' \rangle}{\sqrt{a^2 + |\gamma|^2} \left(|\gamma'|^2 a^2 + |\gamma \times \gamma'|^2 \right)} \gamma \times \gamma'. \tag{9}$$

Here we have abbreviated $x - \gamma(s)$ to γ, $\gamma'(s)$ to γ' and the prime is derivation with respect to s.

More explicitly, regard a single edge with startpoint γ_0 and endpoint γ_1: Then

$$\gamma(s) = \gamma_0 + s(\gamma_1 - \gamma_0),$$

and the cross-product

$$\gamma(s) \times \gamma'(s) = \gamma_1 \times \gamma_0$$

does not depend on s. The generated velocity field at zero is

$$u(0) = \langle \gamma_0, \gamma_1 \rangle \frac{\frac{\|\gamma_0\|^2}{\sqrt{a^2 + \|\gamma_0\|^2}} - \frac{\|\gamma_1\|^2}{\sqrt{a^2 + \|\gamma_1\|^2}}}{a^2 \|\gamma_1 - \gamma_0\|^2 + \|\gamma_1 \times \gamma_0\|^2} \gamma_1 \times \gamma_0, \tag{10}$$

any other point x can be evaluated by replacing γ_i by $\gamma_i - x$. This simple formula is actually surprising: Even for circular filaments the integration of the Biot-Savart formula (3) requires elliptic functions. Also the modified kernel used in [AN05] turns out to be much more expensive to evaluate, it includes angle-computations that require evaluation of the arctan-function.

4.2. Local effects at vertices

The two adjacent edges have no influence at all on the velocity of a vertex. This is roughly equivalent to modelling vortex tubes of thickness equal to the edge length of the polygon. Using this model we would be unable to model thin (and therefore fast) filaments without using excessively many edges for each polygon.

Asymptotic analysis of the velocity field at such a vertex induced by its two adjacent edges (see [Wei06]) reveals the

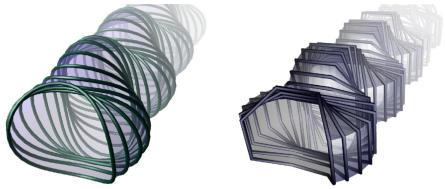

Figure 5: *An oval curve evolving under the smoke ring flow and its discrete analog, from [Hofon].*

following: The contribution of local effects behaves like the smoke ring flow (5) and the resulting equation of motion for a vertex γ_i of a polygonal vortex filament γ is then

$$\dot{\gamma}_i = u(\gamma_i) + \lambda \kappa_i b_i, \tag{11}$$

where u is given by summing up Equation (6) for all filaments using (10), $\kappa_i b_i$ denotes curvature times binormal at γ_i, and λ is constant depending on a. The mathematical notion of curvature is only available for arc-length parametrized polygons, i.e. all edges have equal length. Since the non-local effects quickly destroy any arc-length parametrisation we can not evaluate (11) directly.

On the other hand, it is known that the doubly discrete smoke ring (or Hasimoto) flow [Hof00, Hofon, PSW07] captures excellently the qualitative behaviour of the smooth smoke ring flow (5) for polygons, see Figure 5. Thus we apply it as an integrator of the locally induced velocity term κb.

4.3. The doubly-discrete smoke ring flow

The doubly-discrete smoke ring flow is a discrete evolution equation for closed polygons. This means, given a closed polygon γ, we can compute a new polygon $\tilde{\gamma}$, that corresponds to the time-evolution of the polygon under the smooth smoke ring flow, see [Hof00, Hofon, PSW07]. One step of this evolution is obtained by computing two successive closed *Darboux transforms*. The Darboux transform is obtained from the initial polygon by elementary geometric constructions (Darboux steps). This will be described in Section 4.3.1.

A single step of the evolution depends on two parameters, which determine the corresponding time step of the smooth evolution. In Section 4.3.3 we will describe how to determine the required parameters in order to match the current time step.

4.3.1. Darboux transforms of polygons

We start with the closed polygon γ, with vertices $\gamma_1, \gamma_2, \ldots, \gamma_n$ and edges $S_1 = \gamma_2 - \gamma_1$, $S_2 = \gamma_3 - \gamma_2, \ldots, S_n = \gamma_1 - \gamma_n$. The Darboux transform η of γ is also a polygon with the same number of edges, and same edge lengths. But η is usually not closed, thus it has the vertices $\eta_1, \eta_2, \ldots, \eta_n, \tilde{\eta}_1$ and edges $\tilde{S}_1 = \eta_2 - \eta_1, \tilde{S}_2 = \eta_3 - \eta_2, \ldots, \tilde{S}_n = \tilde{\eta}_1 - \eta_n$. The Darboux transform has two real parameters, namely the *distance* parameter $l > 0$ as well as the *twist* parameter r. The polygon η is obtained by the following procedure:

- Choose a start point η_1 of η, with distance l from γ_1. We denote the distance vector $\eta_1 - \gamma_1$ by lT_1, where T_1 is a unit vector, and l the distance.
- Map η_1 to the next point η_2 of η by a single *Darboux step*, see below. The step depends only on the difference vector lT_1 and the first edge S_1 of γ, and it has two important properties:
  - The distance between η_2 and γ_2 is also l:
    $$\eta_2 - \gamma_2 = lT_2.$$
  - The edge lengths of η and γ are equal:
    $$\|S_1\| = \|\tilde{S}_1\|.$$
- Now that we have obtained η_2 we apply a Darboux step to η_2 to obtain η_3 and so on. More precisely, we can replace the indices 1 and 2 by i and $i+1$, which allows us to determine the next vertex η_{i+1} as soon as we have determined its predecessor η_i. Thus we transport the initial start point η_1 all around the polygon γ, and we finally arrive at $\tilde{\eta}_1$.

A single Darboux step is given as a quaternionic equation, so we identify $\mathbb{R}^3$ with the set of imaginary quaternions $\mathrm{Im}\,\mathbb{H}$. From the current distance vector lT_i and the current edge S_i of γ, we obtain lT_{i+1} as:

$$lT_{i+1} = (-r + lT_i - S_i) lT_i (-r + lT_i - S_i)^{-1}. \tag{12}$$

We can apply this equation iteratively to our initial distance vector lT_1 (between γ_1 and the *start point* η_1 of η) to obtain the distance vector $l\tilde{T}_1$ (between γ_1 and the *end point* $\tilde{\eta}_1$ of η), and we will denote this map by f:

$$f\colon lT_1 \mapsto l\tilde{T}_1. \tag{13}$$

For the doubly-discrete smoke ring flow we need to determine a *closed* Darboux transform. Therefore we need to determine the start point η_1 in such a way that η is closed, i.e. $\tilde{\eta}_1 = \eta_1$. This is the case when the initial distance vector lT_1 is a fixed point of f. To compute a fixed point we use the *power method*, i.e. we start with some initial vector with length l and iterate f until convergence.

4.3.2. A single Darboux step

Now we describe a single *Darboux step* that determines η_{i+1} from η_i. Look at a single quadrilateral with the edges S_i, lT_{i+1}, $-\tilde{S}_i$, $-lT_i$ (Figure 6). Since opposite edges have equal length, it is a parallelogram, folded about one of its diagonals. The folding angle about the diagonal between γ_{i+1} and η_i,

$$D_i = \eta_i - \gamma_{i+1} = lT_i - S_i,$$

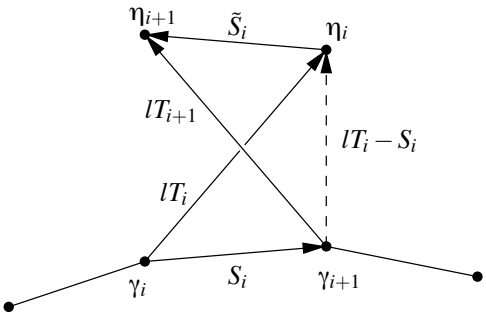

Figure 6: *A polygon γ and a single edge $\tilde{S}_i$ of its Darboux transform η.*

is required to be

$$\alpha_i = 2 \arctan \|D_i\|/r,$$

where the twist parameter r enters. This rule determines η_{i+1} uniquely. In particular, the newly obtained distance vector lT_{i+1} is obtained by an α_i-degree rotation of the previous distance vector lT_i about the diagonal D_i. By adding lT_{i+1} to γ_{i+1}, we have obtained η_{i+1}.

This construction might be implemented by computing the corresponding rotation matrix and apply it to lT_i, but it is much simpler to implement using quaternions, using Equation (12).

4.3.3. Parameters for the Darboux transforms

One step of the doubly-discrete smoke ring evolution is obtained by two subsequent Darboux transforms, the first with parameters l and $+r$, the second with parameters l and $-r$. To determine l and r for a polygon γ with n vertices and total length L, we regard a regular, planar n-gon $\tilde{\gamma}$ with the same total length L and compute its self-induced speed:

$$\tilde{U} = \|u(\tilde{\gamma}_i)\|.$$

Note that $u(\gamma_i)$ (given by Equation (6)) is the same for each vertex $\tilde{\gamma}_i$ and perpendicular to the n-gons plane.

Now we compare $\tilde{U}$ with the analytically known speed of a smooth circular vortex filament with same length:

$$U = \frac{\Gamma}{2L}\left(\ln\frac{4L}{\pi a} - 1\right). \tag{14}$$

The Darboux-transform η of a regular n-gon $\tilde{\gamma}$ is a shifted copy of $\tilde{\gamma}$: It is rotated about the center axis with angle ϕ, and translated along the center axis, with distance d. For the combination of two successive Darboux transforms with parameters l and $\pm r$ the rotations will cancel, and the resulting motion is a pure translation along the center axis, with distance $2d$. The correct speed is obtained when

$$2d = \Delta t\,(U - \tilde{U}).$$

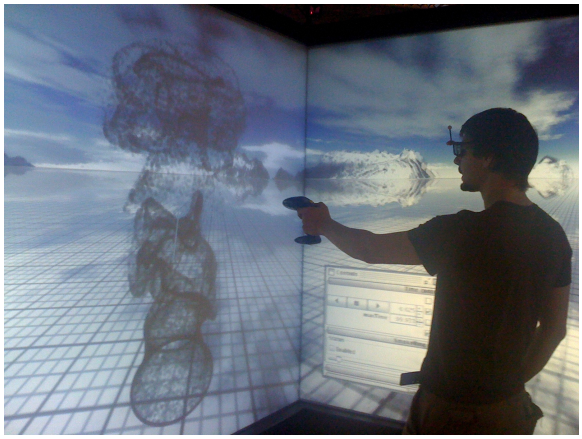

Figure 7: *Interactive smoke simulation in an immersive, distributed environment.*

With a coice of the rotation angle $\phi = 2\pi/n$, the parameters turn out to be

$$l = \sqrt{(L/n)^2 + d^2}, \quad r = d\,\cot(\pi/n). \tag{15}$$

4.4. Summary

We have presented all necessary formulas to compute a single step of the evolution of polygonal vortex filaments. In Equation (10) we have presented the explicit formula to evaluate the velocity field induced by the individual edges. We can use this to directly advect the filament vertices for a given time step using a standard forward ODE integration scheme, for instance the Runge-Kutta scheme RK4. But the resulting motion of the discrete filaments does not adequately reproduce the physical system: It underestimates the locally induced speed of the individual filaments. We account for this by applying a single step of the doubly-discrete smoke ring flow evolution to each of the filaments. Therefore one first needs to determine the twist parameter r and the distance parameter l for the given time step, using the Equations (15) and (14). The step itself is obtained by computing two successive Darboux transforms. This is done by determining a fixed point of the map defined in Equation (13), which is itself defined by a sequence of Darboux steps (12). The fixed point is obtained by applying the power method.

5. Implementation

We will now describe how to implement a fluid simulator using the evolution of discrete vortex filaments. The implementation splits into two independent parts: The evolution of vortex filaments, and the advection of particles. From the evolving filaments, we obtain the current velocity vector field at each time step. We can evaluate this velocity field on the whole of $\mathbb{R}^3$ using the Biot-Savart formula (6). We will

use this velocity vector field for particle advection, using a standard ODE integration scheme.

In our implementation this second part is implemented using a GLSL shader to perform particle advection on a GPU. The filament simulation is implemented in a Java library. The most expensive part of the filament simulation is the advection step which includes evaluations of the induced velocity field at all filament vertices. A significant speed up can be achieved when implementing this part on the GPU, but in our experimental implementation we favor the greater flexibility of a CPU implementation.

For particle advection we transfer the whole set of edges to the GPU. This needs to be done in every time step. The particle positions are also stored on the GPU, but they are only transferred during the initialization of the simulation. This allows to advect a large number of particles compared to the number of filament edges, at interactive rates.

The implementation uses jReality [WGH*09], which provides the infrastructure for interaction and rendering. Applications written in jReality will run also in distributed, immersive virtual environments with head-tracking and tracked input devices, see Figure 7. Interaction with the 3D scene is usually done with a *pointer device*, which is the mouse pointer in a default desktop setup and the 6DOF tracked wand in an immersive environment.

5.1. Filament evolution

Each simulation step of the filament evolution consists of the two steps:

1. Evolve each filament by one step of the doubly-discrete smoke ring evolution, by computing two successive Darboux transforms.
2. Advect the vertices of the filaments according to the evolution Equation (4), using a standard ODE integration scheme. The right hand side is obtained by implementing Equation (10) and summing up over all filament edges.

5.1.1. Implementation of the doubly-discrete smoke ring evolution

To compute one step of the doubly-discrete smoke ring evolution for time step Δt we need to do the following:

1. Determine the length L and number of vertices n of γ.
2. Compute U according to Equation (14).
3. Compute $\tilde{U}$. Construct a reference n-gon with same length as γ and evaluate its own induced speed at a vertex.
4. Compute l and r according to Equations (15).
5. Compute the Darboux-transform η of γ with parameters l and r.
6. Compute the Darboux-transform $\hat{\gamma}$ of η with parameters l and $-r$.

Now $\tilde{\gamma}$ is the required step of the doubly-discrete smoke ring evolution of γ.

We will now explicitly implement the computation of the Darboux-transform. The following pseudocode assumes that vectors in $\mathbb{R}^3$ are implemented by a class `real3` and quaternions are implemented in a class `quat`. Quaternions are constructed by passing the real part as a `double` and the imaginary part as a `real3`. The two classes have all standard operators implemented, `quat.inverse()` gives the inverse of a quaternion. A polygon is stored as a `vector` containing all polygon vertices.

First, we implement a single Darboux step, as described in Equation (12):

```
real3 darboux_step(real3 S_i, real3 lT_i, double r) {
  quat rlT_S = quat(-r, lT_i - S_i);
  quat lT = quat(0, lT_i);
  quat lTnext = rlT_S*lT*rlT_S.inverse();
  return lTnext.imag();
}
```

Then we compute the end vector $l\tilde{T}_1$ for a given start vector lT_1, see Equation (13):

```
real3 monodromy(vector<real3> gamma, real3 lT_1, double r) {
  real3 lT = real3(lT_1);
  for (int i=0; i<n; i++) {
    real3 S_i = gamma[i+1]-gamma[i];
    lT = darboux_step(S_i, lT, r);
  }
  return lT;
}
```

Now we determine a start vector lT that leads to a *closed* Darboux transform. This vector is a fixed point that we determine by applying the power method to the previous method:

```
real3 power_method(vector<real3> gamma, double l, double r) {
  real3 lT = real3(0, 0, l);
  for (int i=0; i<MAX_ITERS; i++) {
    real3 lastLT = real3(lT);
    lT = monodromy(gamma, lT, r);
    if ((lT - lastLT).norm() < EPS) return lT;
  }
  // signal: "did not converge"
}
```

Finally, we construct the closed Darboux transform η using the start vector lT that gives a closed solution:

```
void darboux(vector<real3> gamma,
             vector<real3> eta, double l, double r) {
  real3 lT = power_method(gamma, l, r);
  for (int i=0; i<n; i++) {
    eta[i]=gamma[i]+lT;
    real3 S_i = gamma[i+1]-gamma[i];
    lT = darboux_step(S_i, lT, r);
  }
}
```

Note that in some rare cases the power method might fail to converge (which usually does not happen in our simulations). In this case we just skip the computation of the Darboux transform. To handle these cases also, a more sophisticated method, for instance the periodic QR decomposition [Kre06], needs to be implemented to determine the fixed point.

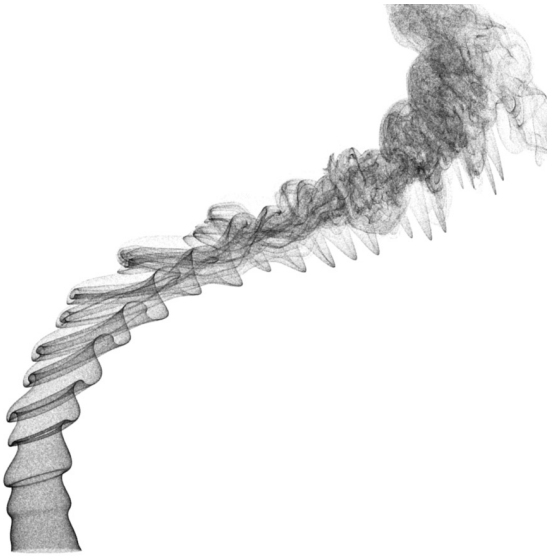

Figure 8: *Control of large scale flow features. The image shows the evolution of a jet, guided along a spiral curve using a discretized vortex tube.*

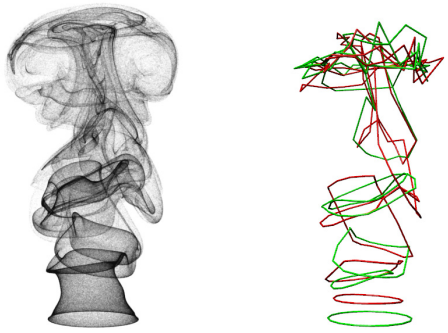

Figure 9: *Simulation of a jet. The left picture shows* 1024^2 *particles, injected into the flow from a circular curve at the bottom, where also the filaments are inserted. The right hand side shows the vortex filaments at the same time of the simulation.*

5.2. Particle advection

Particle advection is performed using the second-order Runge-Kutta scheme. This scheme requires the velocity field (and thus the filaments) only at $t = t_i + \Delta t/2$, other schemes would require more filament positions during one time step, resulting in higher traffic between CPU and GPU memory. The advection step of a particle is implemented as a GLSL fragment shader. The fragment shader iterates all vortex filament edges, transforms them such that the evaluation point is at the origin, evaluates the velocity field generated by the current edge, and sums up.

5.3. Rendering

Our focus is the real-time interactive simulation algorithm, rather than rendering. For this reason we have choosen the cheapest and fastest way to render the marker particles: As plain, unshaded, highly transparent points.

5.4. Interaction

Our implementation contains several modelling tools for interactive design of fluid motion. The tools to design an initial setup:

Filament editor: Closed polygonal vortex filaments can be modelled as an initial vorticity configuration. Each closed polygon has an assiciated strength Γ and a thickness parameter a. Figure 1 has been made using three initial filaments.

Vortex tube: Static vortex filaments can be placed along the cross-sections of a tube, which is modelled by its center curve and a given radius. This is used to model a background flow that guides the overall motion of the fluid, see Figures 4, 8.

Jet emitter: A jet is simulated by frequent emission of circular filaments. This is possible since the originally created vortex sheet rolls up to vortex core structures (Section 3). To break symmetry, a random distortion is applied to each filament. This was used in the Figures 8 and 9, and also below the sphere in Figure 3.

For interaction during the simulation one can create new smoke rings using the mouse or a wand in an immersive environment.

6. Results and discussion

The system we have described allows the interactive simulation of 3D smoke on desktops and also in immersive virtual environments. The system is able to simulate realistic and physically accurate fluid motions using a small number of filaments, as shown in the figures. Using the GPU it is possible to advect a large number of marker particles in real time. Table 1 shows framerates measured on a Intel Core 2 Extreme CPU X9650 3.00GHz with 4 GB RAM and a nVidia GeForce 8800 Ultra GPU. Note that the frame rate is limited to 200 fps by the update rate of the Java AWT event queue.

Because of the physically accurate Biot-Savart law, our approach is independent of scale; scaling the fluid domain will result in a correctly scaled velocity field. In contrast, the method of Angelidis and Neyret requires adjustments of parameters that are not physically meaningful. By the use of the doubly-discrete smoke ring flow we are able to model even thin filaments with a small number of edges.

The resulting simulations are accurate for a limited

Table 1: *Performance in fps inlcuding simulation and rendering.*

	# Edges				
# Particles	8	64	128	256	512
4096	>200	>200	180	115	9.4
16384	>200	>200	148	94	9.0
65536	195	126	90	56	8.1
262144	63	42	33	21.8	5.5
1048576	16.8	11.85	9.7	6.2	2.6

amount of time, even when the filaments incur long edges and sharp cusps, see for instance the filaments shown in Figure 9. From the discussion about energy and the geometry of the smooth filaments in Section 3.3 we conclude that polygons need to be subdivided in order to capture the increasing complexity of the smooth filaments. This will however quickly lead to polygons with excessively many edges and to edge lengths that cause problems to the numerics.

Previously described methods to increase stability for long-time simulations can also be added to our method, although they are not satisfying: An unnatural amount of damping [AN05], and filtering of high frequencies of the filaments [ANSN06]. We believe that in order to implement long-time simulations one has to cope with the increasing complexity of the filaments by handling topology changes due to reconnections.

7. Conclusion and outlook

We have presented a method to simulate smoke that uses the physically correct velocity field. Local effects are captured by the doubly-discrete smoke ring flow. Thus the simulation is physically accurate even for a coarse discretization of the filaments. We have implemented the method for desktop and immersive virtual environments. Using GPU techniques, our solution runs in real time.

Current research aims to introduce arbitrary polygonal meshes as boundary conditions, and to handle filament reconnections.

8. Acknowledgement

We would like to express our gratitude to Peter Schröder, for his support and important comments. This work is supported by the DFG Research Center Matheon.

References

[AK98] ARNOLD V. A., KHESIN B. A.: *Topological Methods in Hydrodynamics*, vol. 125 of *Applied mathematical sciences*. Springer, New York, 1998.

[AN05] ANGELIDIS A., NEYRET F.: Simulation of smoke based on vortex filament primitives. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM, pp. 87–96.

[ANSN06] ANGELIDIS A., NEYRET F., SINGH K., NOWROUZEZAHRAI D.: A controllable, fast and stable basis for vortex based smoke simulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 25–32.

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 209–217.

[Cho90] CHORIN A. J.: Hairpin removal in vortex interactions. *J. Comput. Phys. 91*, 1 (1990), 1–21.

[Cho91] CHORIN A. J.: *Vorticity and Turbulence*, vol. 103 of *Appl. Math. Sci. Ser.* Springer, New York, 1991.

[Cho93] CHORIN A. J.: Hairpin removal in vortex interactions II. *J. Comput. Phys. 107*, 1 (1993), 1–9.

[CLT07] CRANE K., LLAMAS I., TARIQ S.: *GPU Gems 3 - Real-Time Simulation and Rendering of 3D Fluids*. Addison-Wesley, 2007, ch. 30, pp. 633–673.

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (New York, NY, USA, 1996), Springer-Verlag New York, Inc., pp. 61–76.

[EM70] EBIN D., MARSDEN J.: Groups of diffeomorphisms and the motion of an incompressible fluid. *Ann. of Math. 92* (1970), 102–163.

[EVG07] EK L. A., VISTNES R., GUNDERSEN O. E.: Animating physically based explosions in real-time. In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa* (New York, NY, USA, 2007), ACM, pp. 61–69.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 15–22.

[Has72] HASIMOTO H.: A soliton on a vortex filament. *Journal of Fluid Mechanics 51* (1972), 477–485.

[Hof00] HOFFMANN T.: *Discrete curves and surfaces*. PhD thesis, Technische Universität Berlin, 2000.

[Hofon] HOFFMANN T.: Discrete Hashimoto surfaces and a doubly discrete smoke ring flow. In *Lectures on Discrete Differential Geometry*, Bobenko A. I., Schröder P., Sullivan J. M., Ziegler G. M., (Eds.), Oberwolfach Seminars. Birkhäuser, Basel, in preparation. Preprint arXiv:math/0007150v1.

[Jef] JEFFERY G.: Coloured smoke. online. http://sensitivelight.com/smoke2, image 14.

[Kre06] KRESSNER D.: The periodic qr algorithm is a disguised qr algorithm. *Linear Algebra and its Applications 417*, 2-3 (2006), 423 – 433. Special Issue in honor of Friedrich Ludwig Bauer.

[KW05] KRÜGER J., WESTERMANN R.: GPU simulation and rendering of volumetric effects for computer games and virtual environments. *Computer Graphics Forum 24*, 3 (2005).

[LLW04] LIU Y., LIU X., WU E.: Real-time 3d fluid simulation on gpu with complex obstacles. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (Oct. 2004), pp. 247–256.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

[PK05] PARK S. I., KIM M. J.: Vortex fluid for gaseous phenomena. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM, pp. 261–270.

[PSW07] PINKALL U., SPRINGBORN B., WEISSMANN S.: A new doubly discrete analogue of smoke ring flow and the real time simulation of fluid flow. *Journal of Physics A: Mathematical and Theoretical 40*, 42 (2007), 12563–12576.

[Ric91] RICCA R. L.: Rediscovery of Da Rios Equations. *Nature 352* (1991), 561–562.

[RPN00] ROCKLIFF S. H. L., PETER RYAL VOKE, NICOLE JACQUELINE: Three-Dimensional Vortices of a Spatially Developing Plane Jet. *International Journal of Fluid Dynamics 4* (2000), 1–+.

[Saf92] SAFFMAN P. G.: *Vortex Dynamics*. Cambridge University Press, Cambridge, 1992.

[SDR06] SANTE DA RIOS L.: Sul moto d'un liquido indefinito con un filetto vorticoso di forma qualunque. *Rendiconti del Circolo Matematico Palermo 22* (1906), 117–135.

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM, pp. 129–136.

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. 24*, 3 (2005), 910–914.

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

[TLP06] TREUILLE A., LEWIS A., POPOVIĆ Z.: Model reduction for real-time fluids. *ACM Trans. Graph. 25*, 3 (2006), 826–834.

[Wei06] WEISSMANN S.: *Real-time simulation of fluid flow*. Master's thesis, TU Berlin, 2006.

[WGH*09] WEISSMANN S., GUNN C., HOFFMANN T., BRINKMANN P., PINKALL U.: jReality. In *ACM Multimedia 2009 - Open Source Software Competition* (Beijing, P.R. China, 10 2009).

[Yan09a] YANG Q.: Real-time simulation of 3d smoke based on navier-stokes equation. *W. Trans. on Comp. 8*, 1 (2009), 103–112.

[Yan09b] YANG Q.: Real-time simulation of 3d smoke on gpu. In *CISST'09: Proceedings of the 3rd WSEAS international conference on Circuits, systems, signal and telecommunications* (Stevens Point, Wisconsin, USA, 2009), World Scientific and Engineering Academy and Society (WSEAS), pp. 130–134.

[YWH*09] YAN H., WANG Z., HE J., CHEN X., WANG C., PENG Q.: Real-time fluid simulation with adaptive sph. *Computer Animation and Virtual Worlds 20*, 2–3 (2009), 417–426.

[ZSP07] ZHANG Y., SOLENTHALER B., PAJAROLA R.: Gpu accelerated sph particle simulation and rendering. In *SIGGRAPH '07: ACM SIGGRAPH 2007 posters* (New York, NY, USA, 2007), ACM, p. 9.