A Controllable, Fast and Stable Basis for Vortex Based Smoke Simulation
Alexis Angelidis, Fabrice Neyret, Karan Singh, Derek Nowrouzezahrai

To cite this version:

HAL Id: inria-00402106
https://inria.hal.science/inria-00402106
Submitted on 20 May 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Controllable, Fast and Stable Basis for Vortex Based Smoke Simulation

Alexis Angelidis¹ Fabrice Neyret ² Karan Singh ¹ Derek Nowrouzezahrai¹

¹Dynamic Graphics Project, U. of Toronto ²Laboratoire GRA VIR†

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.
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 [ETK05], 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 [ETK05, 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:

\[ w = \nabla \times v \]  
\[ v = \iiint_{V} \frac{(p - x) \times w}{4\pi \|p - x\|^3} \, dx . \]  

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

\[ \frac{dw}{dt} = w \cdot \nabla v . \]  

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 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 \( 3 \times 3 \) matrix of covariance

\[
\frac{1}{n} \sum c_j - \tilde{c} \cdot (c_j - \tilde{c})^T .
\]  

(4)

This matrix’s eigenvalues \( \lambda_\ell \geq \lambda_0 \geq \lambda_c \) correspond to orthogonal unit eigenvectors \( e_\ell \) which define the plane \((\tilde{c},e_\ell)\) 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

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

(5)

The harmonic analysis of each dimension of \( s \) [AW95] defines three Fourier series\(^\dagger\). A new signal \( \tilde{s} \) can be built with a bounded number of Fourier coefficients and the new synthesized curve is obtained with

\[
\tilde{c}'(l) = \tilde{c} + \sum (e_x, e_y, e_z) \cdot s(2\pi l/L) .
\]

(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.

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\(^\ddagger\). The velocity induced by one filament becomes

\[
\tilde{v} = \int_{l \in [0,L]} \gamma \tilde{f}(||\bar{p} - c||^2/\rho^2)(\bar{p} - c) \times \tau \, dl
\]

(7)

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

The scalar \( l \) parameterizes the filament, \( \tau \) is the tangent at \( c \), \( \gamma \) 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 \( v_{\text{total}} \) induced by multiple filaments is the sum of all filaments’ velocities. Smoke particles may be advected directly with the above field \( \tilde{v} \). 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:

\[\text{Geometry Synthesis} \xrightarrow{\text{Advvection}} \xrightarrow{\text{Harmonic Analysis}}\]

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

\(^\dagger\) A change in the origin or axes of the Cartesian coordinate system may introduce edges to the signal of the geometry.

\(^\ddagger\) \( \tilde{f} \) is monotonic, \( C^1 \), numerically very close to \( C^2 \), and numerically almost antisymmetric about 1/2.
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 \( \bar{c} \) gives an average location of the motion, the vector \( e_c \) 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.

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 \( \phi(l) = m \cdot \tau \in [0, 1] \). This function is conservative, since \( \int \phi \cdot \tau \, dl = 0 \). 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 \( e_c \) gives the direction taken by a filament, the filament’s frame can be rotated to align \( e_c \) 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.

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, \( \bar{c}_p \) and \( \bar{c}_n \). The filament’s new arc-length parameter is \( d' = d + || \bar{c}' - \bar{c}_n || \). The new direction is given by the tangent of the curve \( e_c' = \tau(d') \). The constrained centroid \( \bar{c}' \) is a projection of \( \bar{c}_n \) on the plane \( (K(d'), \tau(d')) \). Note that \( e_c' \) and \( e_c \) 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
We propose to

\[ \mathbf{v}_{\text{tornado}} = \sum_{i=1}^{n} \tilde{f}(\|p - \mathbf{c}_i\|^2 / r^2)(p - \mathbf{c}_i) \times \mathbf{c}_i \]  

(8)

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 twisting velocity is measured at a point with the displacement gradient tensor of that step: \( \frac{\partial \mathbf{v}}{\partial \mathbf{x}} \), where \( \mathbf{x} \) is the position at time \( t \). The stretching accumulated during a longer period of time is given by the chain rule

\[ \frac{dx_n}{dx_0} = \frac{dx_n}{dx_{n-1}} \frac{dx_{n-1}}{dx_{n-2}} \cdots \frac{dx_1}{dx_0}, \]

(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 \( \frac{dx_i}{dx_0} \cdot \frac{dx_j}{dx_0} \).

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-
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

\[
n = \left( \frac{\text{d}x_0}{\text{d}x} \right)^{-1} \cdot \mathbf{e}.
\]

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 filament’s 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.

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-
Geometrically, the Biot-Savart law expresses the velocity induced by \( \mathbf{w} \) at \( \mathbf{p} \) as the velocity of simultaneous rotations of angle 
\[
\frac{1}{4\pi \| \mathbf{p} - \mathbf{x} \| ^3}
\]
(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 \( \times \) 4 matrix that multiplies \( \mathbf{p} \), by expressing the cross product as a matrix
\[
v = \int_0^t \frac{-1}{4\pi \| \mathbf{p} - \mathbf{x} \| ^3} \left( \begin{array}{ccc} \mathbf{x} \times \mathbf{w} \\ 0 & 0 & 0 & 0 \end{array} \right) \, dx \cdot \mathbf{p}.
\]

If we call \( M \) the above matrix, it belongs to the Lie algebra \( \mathfrak{se}(3) \), and defines the trajectory of \( \mathbf{p} \) as \( \exp(t \ M) \cdot \mathbf{p} \), and the new position after a time step of length \( \Delta t \) is \( \exp(\Delta t \ M) \cdot \mathbf{p} \) [Gal00], where \( \exp \) is defined as \( \sum_{n=0}^{\infty} \frac{t^n M^n}{n!} \). In the case of one filament with a discrete integral and our fast vortex distribution function, \( M \) becomes
\[
M = \sum_{i=0}^{\infty} -\gamma \left( \frac{\| \mathbf{p} - \mathbf{c} \| ^2 / r^2 \} \right) \begin{bmatrix} \mathbf{c} \times \mathbf{c} \times \mathbf{t} \end{bmatrix}
\]
which provides a more accurate trajectory than (7).

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 can 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

References


**Scene** | **Modeling Settings** | **High-Quality Settings**
--- | --- | ---

<table>
<thead>
<tr>
<th>Happy Boy</th>
<th>AVG # of Motion Samples</th>
<th>AVG # of Particles</th>
<th>AVG # of FPS</th>
<th>PARTICLE SPLITTING</th>
<th>NOISE</th>
<th>AVG # of Motion Samples</th>
<th>AVG # of Particles</th>
<th>AVG # of FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>72</td>
<td>21.71</td>
<td>✓</td>
<td>450</td>
<td>1197</td>
<td>3.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple Benchmark</td>
<td>78</td>
<td>96</td>
<td>18.43</td>
<td>✓</td>
<td>90</td>
<td>168</td>
<td>7.63</td>
<td></td>
</tr>
<tr>
<td>Toxic Fumes vs. the Fly</td>
<td>45</td>
<td>112</td>
<td>18.32</td>
<td>✓</td>
<td>✓</td>
<td>182</td>
<td>3787</td>
<td>0.37</td>
</tr>
<tr>
<td>Noise Benchmark</td>
<td>91</td>
<td>112</td>
<td>12.63</td>
<td>✓</td>
<td>✓</td>
<td>364</td>
<td>3038</td>
<td>2.516</td>
</tr>
<tr>
<td>Genie Lamp</td>
<td>67</td>
<td>2493</td>
<td>12.435</td>
<td>✓</td>
<td>✓</td>
<td>450</td>
<td>73357</td>
<td>0.24</td>
</tr>
<tr>
<td>Forest Fire</td>
<td>120</td>
<td>4875</td>
<td>5.16</td>
<td>✓</td>
<td>✓</td>
<td>336</td>
<td>20250</td>
<td>0.54</td>
</tr>
<tr>
<td>Walkthrough Smoke</td>
<td>395</td>
<td>1280</td>
<td>4.744</td>
<td>✓</td>
<td>✓</td>
<td>750</td>
<td>7040</td>
<td>1.03</td>
</tr>
</tbody>
</table>

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.


© The Eurographics Association 2006.