Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
P. Kry and J. Lee (Editors)

Smoke Sheets for Graph-Structured Vortex Filaments

Alfred Barnat and Nancy S. Pollard

Carnegie Mellon University

Abstract

Smoke is one of the core phenomena which fluid simulation techniques in computer graphics have attempted to capture. It is both well understood mathematically and important in lending realism to computer generated effects. In an attempt to overcome the diffusion inherent to Eulerian grid-based simulators, a technique has recently been developed which represents velocity using a sparse set of vortex filaments. This has the advantage of providing an easily understandable and controllable model for fluid velocity, but is computationally expensive because each filament affects the fluid velocity over an unbounded region of the simulation space. We present an alternative to existing techniques which merge adjacent filament rings, instead allowing filaments to form arbitrary structures, and we develop a new set of reconnection criteria to take advantage of this filament graph. To complement this technique, we also introduce a method for smoke surface tracking and rendering designed to minimize the number of sample points without introducing excessive diffusion or blurring. Though this representation lends itself to straightforward real-time rendering, we also present a method which renders the thin sheets and curls of smoke as diffuse volumes using any GPU capable of supporting geometry shaders.

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

1. Introduction

Vortex filaments provide an efficient means to capture complex fluid flow as compared to traditional grid or particle-based methods. In computer animation, the technique was pioneered by Angelidis and Neyret [AN05, ANSN06] and later refined by Weißmann and Pinkall [PSW07, WP09, WP10], whose contributions include a variational method for vortex reconnection. In nature, vorticity is usually introduced in the form of sheets at object and flow boundaries, but tends to roll up into one-dimensional structures over time. Simulating these filaments directly leads to a naturally compact basis.

The goal of vortex reconnection is to join redundant bundles of filaments that form as regions of opposing vorticity attract each other, reducing the complexity of the simulation over time without significantly impacting its accuracy [WP10]. Whereas Weißmann and Pinkall limit their technique to vortex rings, which form naturally in inviscid fluids and enable the use of Doubly Discrete Smoke Ring Flow for improved self-advection [PSW07], we use a simplified simulation model supporting arbitrary filament structures to enable additional reconnections. Based on this simulation, we develop a set of criteria to preserve accuracy by preventing reconnections which are more likely to produce visible changes in flow.

Since each vortex filament affects the velocity at every point in the simulation, velocity computations required to advect objects are relatively expensive as compared to other fluid simulation techniques. We develop an adaptive, mesh-based smoke surface representation, allowing us to advect less than $\frac{1}{10}$ the number of points a particle-based representation would require to achieve comparable results. Using a similar set of reconnection criteria to those we develop for vortex filaments, we split or join vertices in order to maintain a consistent vertex density across the entire mesh. With a selection of two GPU-based renderers, we can either draw this mesh directly as a thin layer of smoke or render each triangle as a diffuse smoke volume (figure 1).

Figure 1: *A high-resolution smoke plume with 439,934 vertices, simulated using a 719 vertex filament graph with 762 edges.*

2. Background

For general information on simulated fluids in graphics, we refer the reader to a recent survey paper and the references therein [TY09].

Much of the work in fluid animation over the past decade is based on Stam's *Stable Fluids* technique [Sta99], with the addition of vorticity confinement [FSJ01], which attempts to counteract velocity dissipation. Though well suited for confined environments, these techniques require prior knowledge and careful planning for application in open environments, as the entire fluid domain must be discretized. Furthermore, even techniques which apply an adaptive grid spacing [LGF04] are unable to entirely eliminate velocity dissipation due to interpolation.

Smoothed Particle Hydrodynamics (SPH) offers an alternative, where fluid state is associated with mobile particles instead of static grid coordinates [DG96, SF95]. By representing the fluid domain as a collection of discrete particles, this and other Lagrangian methods avoid the need to predetermine and discretize the region of space in which the simulation will take place. However, SPH is best suited for the simulation of a fluid which is mobile within a larger space, such as water surrounded by air, as it is difficult to precisely fill an enclosed space with particles without either allowing holes in regions of high vorticity or reducing the stability of the simulation due to increased pressure.

Localized regions of vorticity have long been recognized as important to the believability of a fluid simulation. A variety of vorticity-preserving techniques have been developed to slow the dissipation of these regions due to velocity diffusion [FSJ01, NSCL08, SRF05]. *Simplical Fluids* [ETK*07] offers a mesh-based solution which preserves circulation by construction. However, even here the ability to resolve these important structures of vorticity remains limited by the sampling resolution of the simulation. Vortex particle formulations offer an alternative analogous to SPH [CCB*08, GLG95, SRF05]. However, similar difficulties in information propagation arise, making them ill-suited for situations in which small timesteps are otherwise unnecessary.

Angelidis and Neyret [AN05] pioneered the use of vortex filaments as a simulation primitive for fluid animation. Since filaments of vorticity naturally form in a turbulent flow, a sparse filament basis is sufficient to approximate complex fluid motion. Pinkall et al. [PSW07] further developed a means to accurately model self-advection of discretized polygonal filaments, and Weißmann and Pinkall [WP10] introduced a physically motivated criteria for vortex reconnection and hairpin removal. These techniques simulate only ring-structured vortices.

The most direct means of simulating smoke is to store density values throughout the fluid domain, usually at the same grid coordinates as the fluid velocity [FSJ01], and to advect these values according to the fluid motion. However, this approach limits the minimum size of smoke features to the grid spacing and can cause aliasing artifacts if this spacing is too large. Smoke particles can simulate fine-scale detail [KW05], but dissipate quickly unless elongated and eventually split [AN05]. Funck et al. [FWTS08] use implied connectivity information to render smoke using sheets of geometry with a particle at each vertex. Alternatively, geometry can be constructed dynamically at each frame by joining adjacent streams of particles [PSCN10]. However, neither of these methods modify particle placements or densities adaptively as the simulation progresses.

Dynamic surface meshes have been used with great success for tracking both fine-scale features on fluid surfaces [WTGT09, TWGT10] and vortex sheets within a fluid, as either the primary means of simulation [Sto06] or to add fine detail to a larger scale Eulerian simulation [PTG12]. A variety of general use surface tracking techniques have been developed [CCB*08, JCNH10], including the El Topo library [BB09b] which has proven effective for smoke animation [BB09a]. These techniques place a particular emphasis on the maintenance of a valid physical interface in the presence of topological changes, which is important for many applications. Such constraints are unnecessary in the case of smoke, however, as the mesh tracks particle density within a fluid.

3. Contributions

We develop a method of filament reconnection which attempts to preserve physical plausibility while allowing for the maximum number of filament reconnections. Our approach is motivated by visual accuracy of the results and designed to preserve a divergence-free vorticity graph.

Rather than enforcing ring-shaped structures of vorticity, we allow arbitrary reconnection between filaments of varying strength, resulting in a directed graph of vortex filaments. This simplifies the overall technique while allowing us to reduce the number of filament edges and vertices beyond previous techniques.

We also explore the use of a triangle mesh with adaptive vertex spacing for smoke surface tracking. We use similar criteria to those of our vortex reconnection technique to control the vertex spacing within this mesh. Our system is able to both dynamically re-mesh within a single layer and arbitrarily reconnect multiple layers of smoke. In addition to the straightforward direct rendering of this mesh, we present a GPU-based rendering technique which smoothes each triangle's density according to an ellipsoidal gaussian kernel.

4. Physical Motivation

Our vortex filament based simulator allows for the construction of any directed graph of filament edges. In regions of complex flow, this structure bears a resemblance to the dense vorticity mesh used by Simplical Fluids [ETK*07], with the key difference being that we advect the vertices themselves rather than the values defined on the mesh.

Due to the smoothing kernel used on each filament's vorticity [WP09], features smaller than this smoothing radius will tend to blend together in regions of complex flow (figure 2). This motivates our reconnection scheme, which primarily enforces a maximum filament length of one half the smoothing radius.

In sparse regions of the vorticity graph, reconnection tends to be straightforward. It is well understood that regions of opposing vorticity will tend to attract each other and merge over time [Saf90, Cho93], and this is the most easily observable form of reconnection which occurs within our system. However, we also allow perpendicular $\perp$ and $+$ shaped junctions to form. Though they do not directly represent the tendency of the filaments to align as they approach each other, the velocity field constructed from their smoothed vorticity will nevertheless demonstrate this effect (figure 2). Furthermore, in the absence of other nearby filaments (i.e. in the case of a simple flow), those joined perpendicularly continue to be drawn towards each other in the direction of opposing vorticity, leading to continued alignment and reconnection along their length.

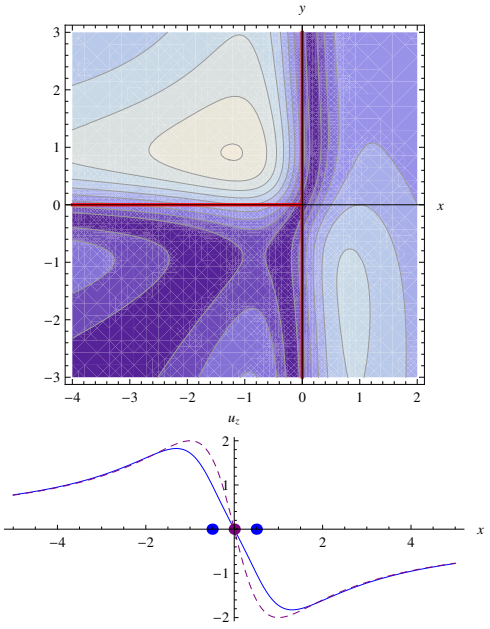

Figure 2: *Velocity fields due to parallel and perpendicular filaments.* ***Top****: The centers of rotation due to perpendicular filaments, shown by the regions of least velocity (dark purple) bend towards each other in the direction of opposing vorticity as they merge.* ***Bottom****: Two parallel filaments of strength 1 produce a velocity field (blue) which converges quickly at distances greater than the smoothing radius to that of a single filament of strength 2 centered between them (purple, dashed). (All filaments have a smoothing radius of 1 and unbounded length. Units are distance, x and y, and velocity,* ***u****, in the simulation space.)*

5. Method

This section describes, in order, our technique for filament reconnection, construction and maintenance of smoke sheets, and smoke rendering. Though intended to complement a vortex filament based simulation, our smoke tracking technique requires only that each vertex in the mesh is advected according to some velocity field, and thus can work with any underlying simulation.

5.1. Filament-based Fluid Simulation

Our fluid simulator is based on the smoothed filament model of Weißmann and Pinkall [WP09]. Though they suggest the use of doubly discrete smoke ring flow [PSW07] to capture more accurate filament self-advection, for simplicity and in order to support non ring-shaped filaments, we instead opt to use only the induced velocity formula derived using the

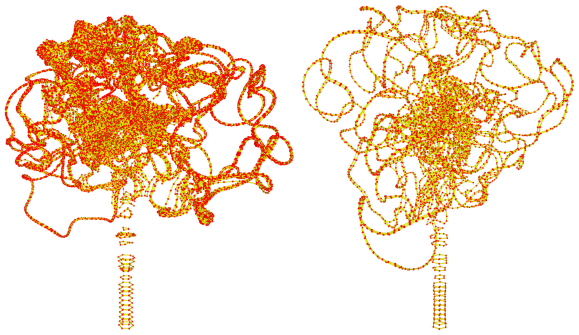

Figure 3: *Without reconnection, thick bundles of vortex filaments form (**left**). With reconnection, the simulation contains less than $\frac{1}{6}$ the number of filaments, though a small amount of viscosity is introduced (**right**).*

Biot-Savart law,

$$\boldsymbol{u}(\mathbf{0}) = (\boldsymbol{\gamma}_i \cdot \boldsymbol{\gamma}_j) \frac{\frac{||\boldsymbol{\gamma}_i||^2}{\sqrt{a^2+||\boldsymbol{\gamma}_i||^2}} - \frac{||\boldsymbol{\gamma}_j||^2}{\sqrt{a^2+||\boldsymbol{\gamma}_j||^2}}}{a^2||\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_i||^2 + ||\boldsymbol{\gamma}_i \times \boldsymbol{\gamma}_j||^2} (\boldsymbol{\gamma}_j \times \boldsymbol{\gamma}_i) \quad (1)$$

[WP09]. Here, $\boldsymbol{u}(\mathbf{0})$ is the velocity at the origin due to a filament of strength 4π from vertices $\boldsymbol{\gamma}_i$ to $\boldsymbol{\gamma}_j$, with smoothing radius a. In order to obtain $\boldsymbol{u}(\boldsymbol{x})$, we simply replace $\boldsymbol{\gamma}_i$ with $\boldsymbol{\gamma}_i - \boldsymbol{x}$. Since this equation produces a divergence free velocity field for each individual vortex filament, the sum of the contributions from all filaments will remain divergence free.

5.1.1. Filament Splitting

Each timestep, we split any vortex filament whose length exceeds its smoothing radius. The new vertex is inserted at a point equidistant to the two end points of the segment, but offset slightly according to the local filament curvature. We compute this offset using a tangent value which we define for each vertex in the filament graph.

The tangent is an average of the direction of each filament attached to the vertex, negated if it has negative strength, weighted by its absolute strength. (A filament of strength Γ from vertex $\boldsymbol{\gamma}_i$ to $\boldsymbol{\gamma}_j$ will behave identically to one of strength $-\Gamma$ from vertex $\boldsymbol{\gamma}_j$ to $\boldsymbol{\gamma}_i$.) We do not normalize these tangents to unit length, as a low magnitude tangent is used to indicate that there is no predominant vorticity near the vertex. We also define vertex strength by taking the length of the sum of all strength vectors (direction scaled by strength) of connected filaments.

$$\boldsymbol{\tau} = \frac{\sum \Gamma_i \hat{\boldsymbol{d}}_i}{\sum |\Gamma_i|} \quad (2)$$

$$\Gamma = ||\sum \Gamma_i \hat{\boldsymbol{d}}_i|| \quad (3)$$

where $\boldsymbol{\tau}$ and Γ are the vertex tangent and strength, and $\boldsymbol{d}_i$ and Γ_i are filament directions and strengths.

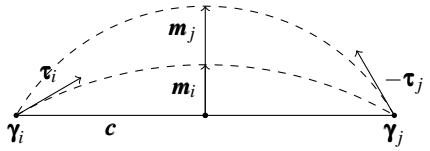

Figure 4: *When splitting the vortex filament $\boldsymbol{c}$ with vertices $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_j$, we compute separate offsets $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ between the midpoint of the filament and the midpoint of the arc constructed according to the tangents at each vertex, $\boldsymbol{\tau}_i$ and $\boldsymbol{\tau}_j$. We then scale each by the magnitude of the corresponding tangent and average the results to obtain the final splitting vertex offset.*

We determine the offset between the physical midpoint of a vortex filament and the point at which we insert the splitting vertex independently based on the tangents at the two endpoints and average the results. The offset corresponding to the first tangent is computed as follows (figure 4):

$$\boldsymbol{c} = \boldsymbol{\gamma}_j - \boldsymbol{\gamma}_i \quad (4)$$

$$\mathbf{v} = (-\boldsymbol{\tau}_i \times \boldsymbol{c}) \times \boldsymbol{c} \quad (5)$$

$$\boldsymbol{m} = \frac{1}{2}||\boldsymbol{\tau}_i||\,||\boldsymbol{c}||\hat{\mathbf{v}} \tan \frac{\arccos(\max(\hat{\boldsymbol{\tau}}_i \cdot \hat{\boldsymbol{c}}, 0))}{2} \quad (6)$$

where $\boldsymbol{m}$ is the offset due to the given tangent. Computation of the second tangent is analogous.

Though unusual, we cap the angle of the arc at 180° in order to place an upper bound on the length of the newly constructed filaments in cases where the tangent is heavily influenced by a filament of opposing vorticity. The splitting operation replaces the original filament with two new filaments of identical strength and radius, connected from the first vertex through the newly created midpoint to the second vertex.

5.1.2. Filament Reconnection

We reconnect filaments by merging nearby vertices in the vorticity graph and adjusting filaments attached to the vertices accordingly. Reconnection can only occur when vertices are within half a smoothing radius of each other, but we also check that the reconnection is in a direction parallel to their tangents and that the tangents themselves are similar. This allows us to reduce detail in regions of complex motion while preserving the behavior of directly opposing vortex pairs.

We implement these constraints as a pair of factors between 0 and 1, reducing the maximum reconnection distance between any given pair of vertices, $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_j$. The first of these reduces reconnections between vertices with differing tangents to compensate for their comparatively larger effect

on local vorticity:

$$\frac{\boldsymbol{\tau}_i \cdot \boldsymbol{\tau}_j + 1}{2}. \tag{7}$$

We then observe that reconnections which simply relocate a vertex along an existing filament will have no effect on the velocity field, whereas reconnections which move a vertex perpendicular to its connected filaments will have the greatest effect. In order to limit the latter, we further scale the reconnection distance by the geometric mean of the dot products between the vertex tangents and the offset between them, weighted by the magnitudes of their tangents (described in the previous section):

$$||\boldsymbol{\tau}_i||\,||\boldsymbol{\tau}_j|| \sqrt{\left| \frac{(\hat{\boldsymbol{\tau}}_i \cdot (\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_i))(\hat{\boldsymbol{\tau}}_j \cdot (\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_i))}{||\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_i||^2} \right|} + (1 - ||\boldsymbol{\tau}_i||\,||\boldsymbol{\tau}_j||). \tag{8}$$

We merge vertices by averaging their positions, weighted by their aggregate strength. We then merge any filaments which become coincident by summing their strengths and averaging their radii and we remove any filaments which connected the old vertices. This procedure ensures that the vorticity graph remains divergence free (i.e. the strength entering any vertex is equal to the strength exiting), as it is equivalent to simply moving the merged vertices to be coincident, modulo any difference in filament radii.

5.2. Smoke Sheets

Our smoke representation consists of an indexed triangle mesh with smoke density stored at each vertex. The density is evenly distributed over the surrounding triangles, so that the density per unit area at the vertex, which translates loosely to absorbance, will change naturally as the area of the triangles changes.

The mesh is seeded from an emitter consisting of a set of fixed vertices stitched to normal, freely moving vertices. As the vertices are advected, the triangle edges connecting the free and fixed vertices become long enough to split and smoke is drawn out of the emitter as a sheet of triangles (figure 5). Though we place the emitters in a ring coincident with the vortex source, the configuration is arbitrary. The smoke mesh can easily be seeded with additional geometry at any point in time, possibly attached to the emitters, which will automatically conform to our constraints within a few timesteps.

As with the vorticity graph, we split and merge vertices in the smoke mesh in order to maintain a consistent level of detail in areas of expansion or constriction and areas where multiple smoke sheets overlap. The basic split and merge operations resemble the mesh repair operations described by Jiao et al. [JCNH10]. However, as any change to the smoke mesh is immediately visible, we place additional restrictions on when this may take place.

5.2.1. Triangle Splitting

Like vortex filaments, triangles are split whenever any of their edge lengths exceed a user-defined threshold, S_d. In addition, we split triangle edges when the angle between the velocities at each vertex exceeds a second user-defined threshold, S_θ, in order to ensure that the triangle mesh can properly track the curvature of nearby vortex filaments. On the occasion that a smoke sheet actually intersects a vortex filament, however, this could lead to a potentially unbounded vertex density, so we do not split an edge if its length is less than a third threshold, S_m. In short, we select an edge containing $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_j$ for splitting if either of the following conditions are met:

$$||\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_i|| > S_d \tag{9}$$

$$\hat{\boldsymbol{u}}(\boldsymbol{\gamma}_i) \cdot \hat{\boldsymbol{u}}(\boldsymbol{\gamma}_j) < \cos S_\theta \quad \text{and} \quad ||\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_i|| > S_m \tag{10}$$

where $\boldsymbol{u}(\boldsymbol{\gamma}_i)$ is the velocity at vertex $\boldsymbol{\gamma}_i$.

To split the edge, we insert a new vertex $\boldsymbol{\gamma}_m$ between $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_j$ and split each triangle containing the edge. We then place $\boldsymbol{\gamma}_m$ at the average of the positions of all connected vertices, weighted by the inverse of their *coverage* (described below) in order to preserve the smoke's silhouette. In this way, we avoid pleating (where the mesh folds in on itself) along triangle edges in areas of constriction, particularly near the smoke source where the discretization tends to follow a regular pattern. Since pleating is not an issue along the edges of the mesh, we avoid any inaccuracies whatsoever when an edge is part of only one triangle, and simply place the new vertex in the middle of the edge.

Our definition of vertex coverage is similar to that of angle defect and thus directly related to the gaussian curvature at the vertex, assuming the vertex is part of a single smoke sheet. The value is simply the sum of all angles containing the vertex, divided by 360°. It provides a measure of the configuration of local geometry surrounding the vertex: vertices surrounded evenly on all sides will have a coverage value near 1, vertices important to the silhouette of the smoke will generally have lower coverage, as they will protrude from the main smoke body, and vertices in more complex configurations may have coverages greater than 1.

In order to complete the split, we must redistribute smoke density within the affected triangles. Briefly, we attempt to ensure that the density per unit area remains constant in all existing vertices. As the insertion of the splitting vertex will generally reduce the total area associated with the surrounding vertices, maintaining a constant density per unit area in these vertices will generate some excess density, which we store in the new vertex. The procedure is discussed in detail in section 5.2.3, as it is common to both splitting and reconnection.

5.2.2. Triangle Reconnection

As with filaments, we perform triangle reconnection by merging adjacent vertices in the smoke mesh. Reconnec-

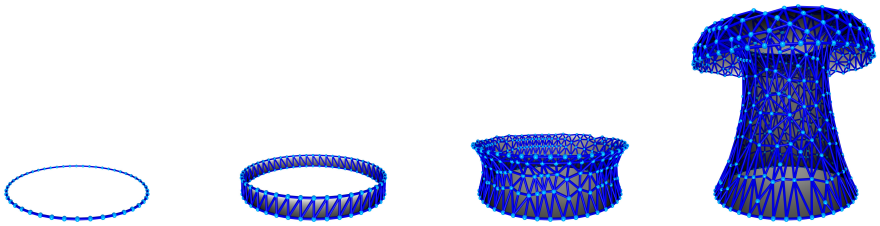

Figure 5: *The smoke source begins as a pair of coincident rings of vertices, one fixed and one advected with the fluid, which form a cylindrical mesh as they are drawn apart. Once the edges are sufficiently long, our triangle splitting procedure begins to insert additional vertices, allowing a smoke sheet to form.*

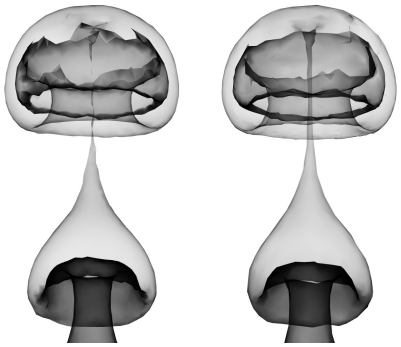

Figure 6: *The adaptive splitting and reconnection distance defined by equations 10, 11, and 12 (**right**) generates smoother smoke and preserves detail better than than a constant distance (**left**), despite using fewer triangles. Detail is otherwise lost in the constricting smoke column and areas of high curvature, such as the curling edges of a smoke plume.*

tion occurs only if the distance between the vertices is at most half of the splitting distance, and we apply several additional constraints to help preserve mesh detail where it has the greatest impact on quality (figure 6).

We first increase edge detail by scaling the reconnection distance by the ratio of coverage (defined in the previous section) between the vertex with lesser coverage and the vertex with greater coverage. If both vertices are in the middle of a sheet, and thus have similar coverage, then this factor will have no effect. However, if one vertex is part of a sheet border and the other is not, the maximum reconnection distance will be cut in half.

We further scale the reconnection distance perpendicular to the local velocity field, in much the same way that we reduce reconnections between parallel vortex filaments:

$$\sqrt{\left|\frac{(\boldsymbol{u}(\boldsymbol{\gamma}_i)\cdot(\boldsymbol{\gamma}_j-\boldsymbol{\gamma}_i))(\boldsymbol{u}(\boldsymbol{\gamma}_j)\cdot(\boldsymbol{\gamma}_j-\boldsymbol{\gamma}_i))}{||\boldsymbol{u}(\boldsymbol{\gamma}_i)||\,||\boldsymbol{u}(\boldsymbol{\gamma}_j)||\,||\boldsymbol{\gamma}_j-\boldsymbol{\gamma}_i||^2}\right|}. \qquad (11)$$

This has the effect of preserving detail in regions of constriction and regions where multiple layers of smoke curl around a vortex.

Finally, we add a constraint to maintain consistency with triangle splitting, as expressed in equation 10:

$$\hat{\boldsymbol{u}}(\boldsymbol{\gamma}_i)\cdot\hat{\boldsymbol{u}}(\boldsymbol{\gamma}_j) > \cos\frac{1}{2}S_\theta \quad \text{or} \quad ||\boldsymbol{\gamma}_j-\boldsymbol{\gamma}_i|| < \frac{1}{2}S_m. \qquad (12)$$

Though reconnection constraints need not be mirrored by splitting constraints, all splitting constraints must be mirrored by similar reconnection constraints so that a vertex is not reconnected immediately after being split, causing loss of detail and potential popping.

If all constraints are met, we merge vertices by averaging their positions, weighted by their respective ratios of area to coverage. This will tend to place the new vertex such that it is surrounded by a roughly symmetric area on all sides, while placing it somewhat closer to areas of low coverage.

Before moving triangles from the old vertices to the new vertex, we determine the maximum change in surface normal between all affected triangles due to the reconnection. As significant changes in surface normal may produce significant changes in effective absorption from the viewing direction, we cancel the reconnection if the angle between the old and new surface normals of any triangles are beyond a user-defined threshold. (In practice, we found a value of 30° provided a good balance between quality and complexity.)

If the reconnection is successful, we remove any degenerate triangles and redistribute the vertex densities, this time seeding the pool of extra density with the combined density of the reconnected vertices.

5.2.3. Density Redistribution

When splitting and reconnecting triangles, we attempt to maintain a constant density per unit area at each vertex. We first attempt to maintain this ratio in all vertices neighboring the reconnected or splitting vertex, and place the remaining density in the new vertex. This method of redistribution naturally preserves total smoke density.

We begin by adjusting the smoke density in each neighbor vertex to maintain the ratio of density to area ($\frac{1}{3}$ the area

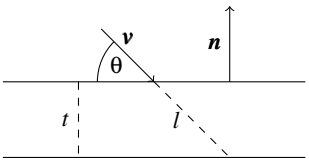

Figure 7: *In our thin-sheet renderer, we treat each triangle as a thin layer of evenly dispersed smoke with thickness t approaching zero. Given the absorption of the sheet when viewed from the normal direction n, we compute the absorption at the ray's angle of incidence (equation 13).*

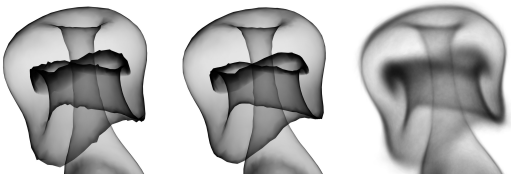

Figure 8: *We remove sharp angles from the smoke (**left**) using a Laplacian smoothing technique (**center**). Volumetric rendering further disguises the underlying mesh (**right**).*

of all triangles containing the vertex), storing the excess for later use. Though reconnection will generally result in negative excess, we also add the density from both reconnected vertices to the pool. If the excess is positive, we simply store it in the new vertex. If there is a shortage, we store no density in the new vertex and remove density from all other vertices proportional to their current total density.

In this manner, we avoid heuristically precomputing the density that should be stored in the new vertex. In the case of reconnection in particular, it is not straightforward to compute this value based on the available information. If the reconnection occurs within a flat sheet, the average ratio of density to area, weighted in the same manner as the average position, should be maintained from the original two vertices. However, when sections from different layers of smoke merge, the new ratio of density to area should be approximately the sum of the old ratios, in order to maintain the same absorbance when viewed from above. In both cases, all neighboring vertices should maintain the same absorption, and thus the same ratio of density to area. By simply enforcing this constraint, reconnections both within and between sheets are handled appropriately.

5.3. Rendering

We develop a pair of rendering techniques: a fast thin-sheet approach which simply draws the mesh as a set of translucent triangles and a more realistic volumetric approach which renders each triangle as a smoothed region of smoke density. Both are implemented entirely on the GPU, with the CPU providing nothing more than vertex locations and either absorption values or densities and blurring radii.

5.3.1. Thin-Sheet

The thin-sheet renderer assumes that each triangle represents a volume of smoke with thickness approaching zero. We define the density per unit area at each vertex, $\frac{3d}{\sum_i a_i}$ where d is smoke density and a is triangle area, to be a measure of the absorbance, α, of a triangle when viewed from above. i.e. if the triangle has thickness t, the absorbance within the triangle volume is $\frac{\alpha}{t}$.

If the view ray travels distance l through the triangle volume, the total absorbance along that path is $\frac{\alpha}{t}l$. Thus, when viewed from an angle θ, the absorbance of the triangle is $\frac{\alpha}{t}t|\csc\theta|$, or given triangle normal $\boldsymbol{n}$ and view ray $\boldsymbol{v}$ (figure 7):

$$\alpha|\csc\theta| = \frac{\alpha}{|\boldsymbol{n}\cdot\boldsymbol{v}|}. \tag{13}$$

Since each vertex may be part of several triangles, we distribute density over the connected faces and combine the absorption contributions due to each face. We use a harmonic mean weighted by triangle area to average the values, as it produces a smoother appearance and simplifies the equation:

$$\frac{\sum_i a_i}{\sum_i a_i \frac{|\boldsymbol{n}_i\cdot\boldsymbol{v}|}{\alpha}} = \frac{3d}{\sum a_i|\boldsymbol{n}_i\cdot\boldsymbol{v}|}. \tag{14}$$

We pass these absorbance values to a simple OpenGL shader, which interpolates them and computes the proportion of background light absorbed at each pixel:

$$\exp(-\alpha). \tag{15}$$

Since we are rendering non-reflective black smoke, the triangle ordering is unimportant.

We optionally apply laplacian smoothing to the vertices in the smoke mesh before rendering each frame (figure 8), substituting each vertex location for the average of the locations of its neighbors, weighted by the inverse of the vertex coverage. This weighting reduces the degree to which the smoothing constricts the smoke's silhouette.

5.3.2. Volumetric Rendering

The volumetric renderer smoothes the smoke density at each point in a triangle over the surrounding region, both enhancing the realism of the resulting animation and helping to obscure the underlying smoke mesh at close viewing distances (figure 8). The majority of the work is done in a shader, based on the position, smoke density, blurring radius, and thickness at each vertex.

The blurring radius, σ_b, is simply the average of the triangle heights (i.e. the distance from a vertex to the opposite side of the triangle) across all vertices of all triangles

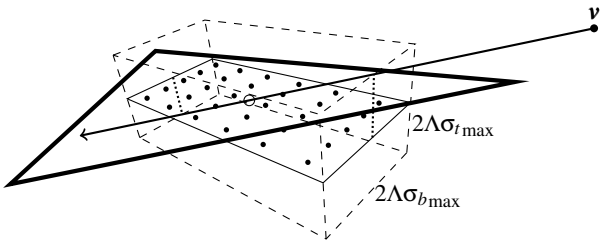

Figure 9: *Our volumetric renderer samples absorbance due to a triangle along view ray* $\boldsymbol{v}$ *at grid points within the triangle plane. Samples are taken within a bounding box of height, perpendicular to the triangle, based on the maximum thickness within the triangle* $\sigma_{t\max}$ *and width, perpendicular to the view ray, based on the maximum blurring radius within the triangle* $\sigma_{b\max}$ *containing all points within the triangle which could contribute density along the path of the view ray under the sampling cutoff* Λ.

containing the vertex in question, scaled by a user-defined constant. It defines the degree of smoothing within the triangle plane, and helps to obscure triangle boundaries without blurring the smoke's silhouette.

The thickness, σ_t, assumes a constant volumetric smoke density throughout the mesh. However, as an unbounded triangle thickness can lead to artifacts, we define this volumetric density to be $\frac{k_d}{k_t}$, where k_t is the maximum thickness and k_d is the density per unit area at which the thickness will be half that:

$$\sigma_t = k_t \left(1 - 2^{-\frac{3d}{k_d \sum a_i}}\right). \tag{16}$$

We smooth the smoke density at each point $\boldsymbol{p}$ of a triangle according to a normalized 3D ellipsoidal Gaussian kernel, consisting of the convolution of a 2D Gaussian of variance $\sigma_b^2(\boldsymbol{p})$ within the triangle plane and a 1D Gaussian of variance $\sigma_t^2(\boldsymbol{p})$ perpendicular to the triangle plane. The shader ray-traces each pixel of each triangle by sampling a set of points within the triangle plane that could contribute density to the path of the view ray (figure 9) and accumulating the contributions from all points that lie within the triangle.

Since this distribution has infinite support, we define a cutoff at Λ standard deviations, beyond which the kernel value is assumed to be 0. (In practice, a cutoff of 2.5 standard deviations works well.) We use this value to limit the sampling region (figure 9) and to discard samples which would have no density contribution given the cutoff, though we do not apply the cutoff when computing absorbance due to each sample that does contribute density.

6. Results

Our vortex filament reconnection method is able to produce accurate results while significantly reducing the number of

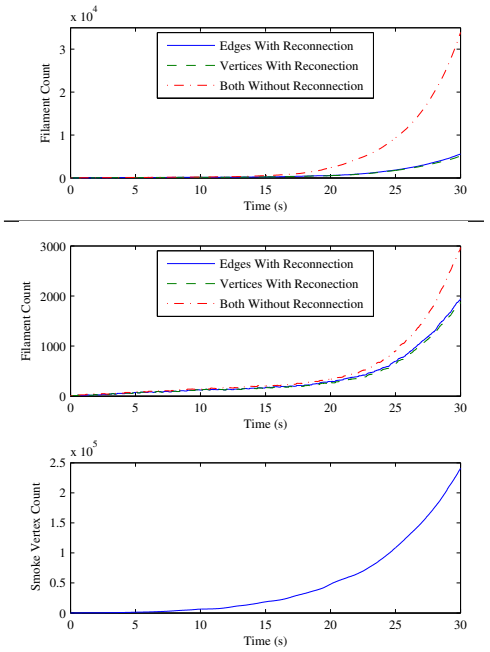

Figure 11: ***Top****: Complexity of the simulation in figure 3.* ***Bottom****: Complexity of the simulation in figure 10. Since this simulation is considerably less turbulent, vortex reconnections have less of an impact*

simulated filaments. In figure 3, we compare the results of otherwise identical simulations with and without reconnection (reconnection is allowed between adjacent vertices within a ring in both cases). Without reconnection, the 1,800 frame simulation completes in 11,778 seconds, and results in 33,881 filaments and vertices. With reconnection, the simulation takes only 186 seconds and results in 5,576 filaments and 5,074 vertices (figure 11). This is a reduction to about 2.5% of the original simulation complexity, measured by the total number of velocity computations per timestep.

The simulation is robust, preserving smoke density perfectly and introducing little additional turbulence through reconnections (figure 3). Despite the constant vortex and smoke reconnections, there is no noticeable drift in vertex locations over time (figure 5).

Our smoke tracking and rendering method is able to produce high quality results, accurately tracking thin sheets of smoke using a limited number of mesh vertices (figure 1). With our triangle splitting and reconnection criteria, we are able to preserve more smoke detail while using fewer triangles (figure 6), allowing us to achieve results of similar quality to those demonstrated by Weißmann and Pinkall [WP10] while using roughly $\frac{1}{10}$ the number of tracking locations (figure 10). The additional complexity of our dynamic vertex density criteria does add some computational overhead, requiring 400 seconds to complete a 900 frame animation,

Figure 10: *A snapshot of the 1,500th frame of a 30 second, 1,800 frame simulation showing the blurred density (**left**), vortex filaments and smoothed density (**center**), and triangle edges (**right**). The smoke mesh contains 108,711 vertices, and the vorticity graph contains 656 vertices and 694 edges.*

as opposed to 300 with a constant density. However, as this overhead is constant in the number of vortex filaments, it is dominated by velocity computation in longer simulations.

Though our volumetric renderer is slower than real-time for smoke meshes of any significant size, the complexity is linear in the number of triangles, and thus will always eventually be exceeded by simulation time. After one second of simulation at high output resolution and moderate smoke vertex density (figure 10), rendering takes approximately 19 times longer than simulation, or 0.079 seconds as opposed to 0.0041 seconds per frame. However, by the end of the simulation, rendering takes 38 seconds while the simulation itself takes 120 seconds per frame. Cumulatively, 21,827 seconds are spent on simulation and 13,007 on rendering for the 30 second, 1,800 frame animation.

7. Discussion

In this paper, we introduce both an extension to existing vortex filament based fluid simulation techniques and a smoke tracking and rendering system designed to reduce the number of velocity computations required to achieve a given level of detail over commonly used particle-based techniques. Our fluid simulator is able to retain a high degree of physical realism while significantly reducing the total number of filaments through reconnection, and our smoke tracking technique is able to preserve thin, sheet-like formations using far fewer points than would be required by existing particle-based systems. Coupled with the volumetric renderer, our system is capable of producing highly detailed, realistic smoke animations using a limited number of filament and smoke vertices.

Though we have shown that a visually inspired approach to reconnection can lead to plausible results, further improvement may be possible by directly bounding the change in velocity due to reconnection, much as Weißmann and Pinkall [WP10] do in cases where the difference in vorticity takes the form of a closed filament loop.

Looking at our smoke tracking system, we may be able to further improve results by tracking additional information in the smoke mesh. For instance we could more accurately distribute density during rendering if we knew that one triangle represented two recently reconnected sheets while others, sharing a vertex, were part of only one of those sheets. Separately, as an extension to volumetric rendering, we might track density dissipation at each vertex, increasing triangle thicknesses and blurring radii over time.

Both filament-based fluids and mesh-based smoke have the potential to enable styles and effects which are not well supported by other simulation and rendering techniques. With the exception of volumetric rendering, the greatest potential may be for real-time applications. Each technique degrades well in quality with reduced complexity and, more importantly, the complexity depends only on the number of features currently being simulated, not the total capacity of the simulation environment.

References

[AN05] ANGELIDIS A., NEYRET F.: Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 87–96. doi:10.1145/1073368.1073380. 1, 2

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

URL: http://portal.acm.org/citation.cfm?id=1218064.1218068. 1

[BB09a] BROCHU T., BRIDSON R.: Animating smoke as a surface. Posters and Demos, 2009. 2

[BB09b] BROCHU T., BRIDSON R.: Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput. 31*, 4 (June 2009), 2472–2493. URL: http://dx.doi.org/10.1137/080737617, doi:10.1137/080737617. 2

[CCB*08] CHATELAIN P., CURIONI A., BERGDORF M., ROSSINELLI D., ANDREONI W., KOUMOUTSAKOS P.: Billion vortex particle direct numerical simulations of aircraft wakes. *Computer Methods in Applied Mechanics and Engineering 197*, 13-16 (2008), 1296 – 1304. URL: http://www.sciencedirect.com/science/article/pii/S0045782507004574, doi:10.1016/j.cma.2007.11.016. 2

[Cho93] CHORIN A. J.: Hairpin removal in vortex interactions ii. *J. Comput. Phys. 107* (July 1993), 1–9. doi:10.1006/jcph.1993.1120. 3

[DG96] DESBRUN M., GASCUEL M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *In Computer Animation and Simulation '96 (Proceedings of EG Workshop on Animation and Simulation* (1996), Springer-Verlag, pp. 61–76. 2

[ETK*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26* (January 2007). doi:10.1145/1189762.1189766. 2, 3

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 15–22. doi:10.1145/383259.383260. 2

[FWTS08] FUNCK W. V., WEINKAUF T., THEISEL H., SEIDEL H.-P.: Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics 14* (2008), 1396–1403. doi:10.1109/TVCG.2008.163. 2

[GLG95] GAMITO M. N., LOPES P. F., GOMES M. R.: Two-dimensional simulation of gaseous phenomena using vortex particles. In *In Proceedings of the 6th Eurographics Workshop on Computer Animation and Simulation* (1995), Springer-Verlag, pp. 3–15. 2

[JCNH10] JIAO X., COLOMBI A., NI X., HART J.: Anisotropic mesh adaptation for evolving triangulated surfaces. *Eng. with Comput. 26*, 4 (Aug. 2010), 363–376. URL: http://dx.doi.org/10.1007/s00366-009-0170-1, doi:10.1007/s00366-009-0170-1. 2, 5

[KW05] KRÜGER J., WESTERMANN R.: Gpu simulation and rendering of volumetric effects for computer games and virtual environments. *Computer Graphics Forum 24*, 3 (2005), 685–693. doi:10.1111/j.1467-8659.2005.00893.x. 2

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23* (Aug. 2004), 457–462. URL: http://doi.acm.org/10.1145/1015706.1015745, doi:http://doi.acm.org/10.1145/1015706.1015745. 2

[NSCL08] NARAIN R., SEWALL J., CARLSON M., LIN M. C.: Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph. 27* (December 2008), 166:1–166:8. doi:10.1145/1409060.1409119. 2

[PSCN10] PARK J., SEOL Y., CORDIER F., NOH J.: A smoke visualization model for capturing surface-like features. *Computer Graphics Forum 29* (2010), 2352–2362. doi:10.1111/j.1467-8659.2010.01719.x. 2

[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* (2007), 12563–12576. doi:10.1088/1751-8113/40/42/S04. 1, 2, 3

[PTG12] PFAFF T., THUEREY N., GROSS M.: Lagrangian vortex sheets for animating fluids. *ACM SIGGRAPH 2012 Papers* (2012). 2

[Saf90] SAFFMAN P. G.: A model of vortex reconnection. *Journal of Fluid Mechanics 212* (1990), 395–402. 3

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 129–136. doi:doi.acm.org/10.1145/218380.218430. 2

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics 24* (2005), 910–914. doi:10.1145/1073204.1073282. 2

[Sta99] STAM J.: Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 121–128. doi:10.1145/311535.311548. 2

[Sto06] STOCK M. J.: *A Regularized Inviscid Vortex Sheet Method for Three Dimensional Flows With Density Interfaces*. PhD thesis, The University of Michigan, 2006. 2

[TWGT10] THÜREY N., WOJTAN C., GROSS M., TURK G.: A multiscale approach to mesh-based surface tension flows. *ACM Trans. Graph. 29* (July 2010), 48:1–48:10. URL: http://doi.acm.org/10.1145/1778765.1778785, doi:http://doi.acm.org/10.1145/1778765.1778785. 2

[TY09] TAN J., YANG X.: Physically-based fluid animation: A survey. *Science in China Series F-Information Sciences 52*, 5 (2009), 723–740. doi:10.1007/s11432-009-0091-z. 2

[WP09] WEISSMANN S., PINKALL U.: Real-time interactive simulation of smoke using discrete integrable vortex filaments. In *VRIPHYS* (2009), pp. 1–10. doi:10.2312/PE/vriphys/vriphys09/001-010. 1, 3, 4

[WP10] WEISSMANN S., PINKALL U.: Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph. 29* (July 2010), 115:1–115:12. doi:10.1145/1778765.1778852. 1, 2, 8, 9

[WTGT09] WOJTAN C., THÜREY N., GROSS M., TURK G.: Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 76:1–76:10. URL: http://doi.acm.org/10.1145/1576246.1531382, doi:http://doi.acm.org/10.1145/1576246.1531382. 2