Smoke Sheets with Arbitrary Reconnection for Filament-based Simulation

Alfred Barnat[*] Nancy S. Pollard[†]

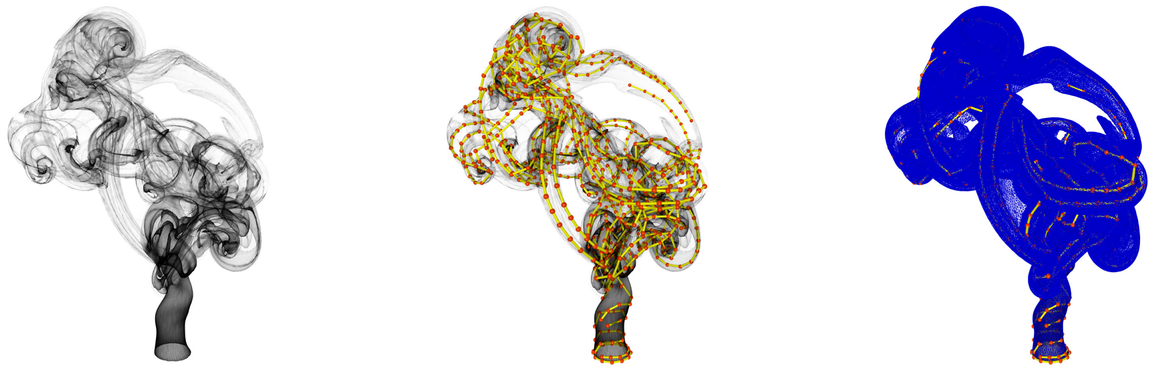

Figure 1: *A smoke plume simulated using our method (**left**). Fluid motion is simulated using vortex filaments (**center**) and the smoke surface is tracked using a triangle mesh (**right**).*

Abstract

Smoke is one of the core phenomena which fluid simulation techniques in computer graphics have attempted to capture. Its behavior is well understood mathematically, and accurate smoke simulation can greatly enhance the realism of computer generated effects. Though much of the work related smoke simulation thus far has used Eulerian grid-based representations of fluid velocity, 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 the entire simulation space. We build upon previous work which merges adjacent rings of filaments in order to minimize the total number of filaments by allowing filaments to form structures other than rings and allowing arbitrary reconnection. To complement this technique, we also introduce a method for smoke rendering designed to minimize the number of sample points without introducing excessive diffusion or blurring. This rendering technique advects a mesh representation of the smoke surface, thus effectively preserving the appearance of thin sheets and curls.

1 Introduction

The majority of smoke simulation methods take an Eulerian approach to the problem [Tan and Yang 2009; Fedkiw et al. 2001; Stam 1999]. As the fluid which drives the smoke's behavior, air, occupies the entire volume of the simulation space, the characteristic space filling grid or mesh of an Eulerian discretization is a natural choice of basis. One of the major advantages of grid-based methods lies in their ability to construct and preserve a divergence free flow through the use of a robust projection step. However, grid-based simulators remain unable to capture fine-scale vortices smaller than the mesh resolution, leading to inherent diffusion of the fluid velocity.

Lagrangian, particle-based simulations offer an alternative. Rather than discretizing the equations of motion into a grid of static cells, they instead associate mass and velocity with moving particles [Stam and Fiume 1995]. Though features can be defined anywhere in the simulation space, rather than only at fixed grid cells, the granularity of the simulation is still limited by the particle density. Furthermore, global optimizations, such as the divergence-eliminating projection step, are difficult to perform on a dynamic set of particles [Cummins and Rudman 1999].

Vortex filaments offer a compromise between these two techniques [Angelidis et al. 2006; Pinkall et al. 2007; Weißmann and Pinkall 2010]. Though Lagrangian in nature, they define the fluid velocity through a series of filaments of vorticity rather than through particles of momentum. This obviates the need for a projection operation, since any vorticity field defines a divergence-free velocity field. Additionally, as a Hamiltonian system, these filament-based simulations are inherently energy preserving.

This method differs from traditional momentum-based particle and grid simulations in that each element of the basis, a vortex filament, contributes velocity to the entire simulation space rather than just a localized region. This means that there is no boundary to the simulation, and thus no related artifacts. However, it increases the computational complexity of the simulation, since each vortex filament is advected by the combined velocity contributions of all other filaments, and makes the simulation of object boundaries significantly more complex [Weißmann and Pinkall 2010].

Despite these shortcomings, vortex filaments offer a compelling alternative to grid or particle based simulations. Only a small number of filaments are required to produce a visually compelling animation, since the sheets of vorticity that form as a fluid is disturbed naturally tend to roll up into filament-like structures. Furthermore, the sparsity of this representation allows the unique possibility of precise artistic control over the very basis of the simulation itself, rather than through more traditional high-level interfaces [Angelidis et al. 2006].

We extend previous work on vortex filaments by relaxing the requirement that filaments exist only as a set of distinct, closed loops, and taking advantage of the additional flexibility to support arbi-

[*]e-mail: abarnat@cs.cmu.edu
[†]e-mail: nsp@cs.cmu.edu

trary reconnection. We further develop a set of heuristics designed to preserve physical plausibility while minimizing the number of vortex filaments present in the simulation.

To complement this technique, we introduce a method for smoke rendering designed to minimize the number of velocity calculations to be performed at each timestep. By simulating sheets of smoke instead of individual disconnected particles, we are able to vastly reduce the number of points being advected. We apply a method similar to what we use for filament reconnection to control splitting and reconnection of edges both within and between sheets. By varying the density of particles across each smoke sheet, we are able to preserve fine details while maintaining an overall low particle count.

2 Background

Much of the work in fluid simulation for computer graphics over the past decade is based on Stam's *Stable Fluids* technique [1999], with the addition of vorticity confinement [Fedkiw et al. 2001] in order to counteract velocity dissipation. For general information on simulated fluids in graphics, we refer the reader to a recent survey paper and the references therein [Tan and Yang 2009]. 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, simulation detail is limited by the predetermined grid spacing, preventing the creation of small-scale vortical flows, and leading to the diffusion of velocity and density values over time.

Smoothed Particle Hydrodynamics (SPH) offers an alternative, where fluid state is associated with mobile particles instead of static grid coordinates [Desbrun and paule Gascuel 1996; Stam and Fiume 1995]. 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 space. However, SPH is best suited for the simulation of a fluid which is mobile within a larger space, such as water surrounded by air (in which case the air is often assumed to exert no force on the water, and is simply not simulated).

Localized regions of vorticity have long been recognized as important features for the believability of a fluid simulation. A variety of vorticity-preserving techniques have been developed to help ensure that these regions do not dissipate unnaturally quickly due to velocity diffusion [Fedkiw et al. 2001; Selle et al. 2005; Narain et al. 2008]. *Simplical Fluids* [Elcott et al. 2007] offers a mesh-based technique which preserves vorticity by construction. However, even here the ability to resolve these important structures of vorticity remains limited by the sampling resolution of the simulation. Vorticity particle formulations offer an alternative analogous to SPH [Selle et al. 2005; Chatelain et al. 2008]. However, similar difficulties arise in their use to drive smoke motion.

Angelidis and Neyret pioneered the use of vortex filaments as a simulation primitive [Angelidis and Neyret 2005]. Since filaments of vorticity naturally form in a turbulent flow, a sparse basis is sufficient to approximate complex fluid motion. Doubly discrete smoke ring flow further developed a means to accurately model self-advection of discretized polygonal filaments [Pinkall et al. 2007], and Weißman et al. introduced a physically motivated criteria for vortex reconnection and hairpin removal [2010]. These techniques generally simulate only ring-structured vortical flows.

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 [Fedkiw et al. 2001]. However, this limits the minimum size of fluid features to the grid spacing, and can cause aliasing artifacts if this spacing is too large. Smoke particles can simulate fine-scale detail, but dissipate quickly [Krger and Westermann 2005]. Funck et al. use implied connectivity information to render smoke using sheets of geometry with a particle at each vertex [2008]. Alternatively, geometry can be constructed dynamically at each frame by joining adjacent streams of particles [Park et al. 2010]. However, neither of these methods modify particle placements or densities once they have been spawned.

3 Contribution

We develop a method of filament reconnection which preserves physical plausibility while allowing for the maximum number of filament reconnections. Our approach is motivated both visually and by the desire to preserve fluid energy.

Rather than enforcing ring-shaped structures of vorticity, we allow arbitrary reconnection between filaments of varying strength, resulting an a directed graph of vortex filaments. In addition to enabling greater flexibility in reconnections, this also makes it possible to obtain a ratio of filaments to vertices greater than 1:1.

Furthermore, we 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, enabling both dynamic re-meshing within a single layer, and arbitrary reconnection between multiple layers of smoke.

4 Method

Though they are intended to complement each other, our vortex-filament based simulation and mesh based smoke tracking and rendering system work independently. Our smoke renderer requires only that each vertex is advected according to the fluid simulation.

4.1 Filament-based Fluid Simulation

Our fluid simulator is based on the smoothed filament model of Weißmann and Pinkall [2009]. Though they suggest the use of doubly discrete smoke ring flow [Pinkall et al. 2007] 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 Biot-Savart law,

$$u(0) = (\gamma_i \cdot \gamma_j)\frac{\frac{||\gamma_i||^2}{\sqrt{a^2+||\gamma_i||^2}} - \frac{||\gamma_j||^2}{\sqrt{a^2+||\gamma_j||^2}}}{a^2||\gamma_j - \gamma_i||^2 + ||\gamma_i \times \gamma_j||^2}||\gamma_j \times \gamma_i|| \qquad (1)$$

[Weißmann and Pinkall 2009]. Here, γ_i and γ_j are the starting and ending points of the filament, and a is its smoothing radius. In order to obtain $u(x)$, simply replace γ_i with $\gamma_i - x$.

4.1.1 Filament Splitting

For each vortex filament, we maintain a target segment length equivalent to its smoothing radius. Though our simulator tracks each filament's radius individually and is technically capable of handling filaments of multiple radii simultaneously, we spawn filaments with only a single constant smoothing radius.

Splitting is straightforward. We simply split a filament segment whenever its length exceeds its radius. The new vertex is inserted at a point which is equidistant to the two end points of the segment, but offset slightly according to the average direction of all filaments previously attached to these two endpoints. In order to facilitate

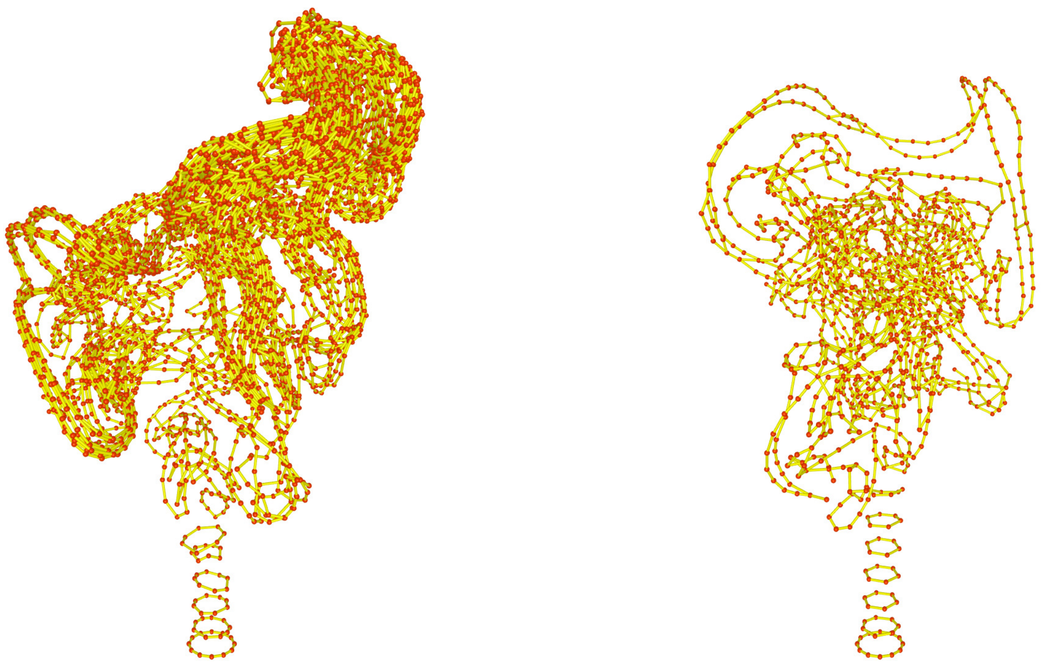

Figure 2: *Without reconnection, thick bundles of vortex filaments form (**left**). With reconnection, the simulation contains $\frac{1}{3}$ the number of filaments, though a small amount of viscosity is introduced (**right**).*

this computation, we track a tangent value for each vertex in the filament graph, which is updated automatically as needed.

The tangent is computed as an average of the directions of each filament attached to the vertex, weighted by their strengths. We store both negative and positive filament strengths, though it is worth noting that a filament of strength Γ from vertex γ_i to γ_j is equivalent to a filament of strength $-\Gamma$ from vertex γ_j to γ_i. We do not normalize these tangents after averaging their values, as we use their magnitude when computing filament midpoints during splitting as a measure of agreement on the vorticity direction at each vertex. Vertex radii are computed analogously, and aggregate strength values are computed by summing each filament's direction multiplied by its strength and taking the length of the resulting vector.

$$\tau = \frac{\sum d_i \Gamma_i}{\sum \Gamma_i} \tag{2}$$

$$r = \frac{\sum r_i \Gamma_i}{\sum \Gamma_i} \tag{3}$$

$$\Gamma = || \sum d_i \Gamma_i || \tag{4}$$

, where τ, r, and Γ are the vertex tangent, radius, and aggregate strength, and d_i, Γ_i, and r_i are filament directions, strengths, and radii.

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 of the two endpoints and average the results. The midpoint is computed as if the filament is an arc, with our previously computed tangents at the end points (figure 3),

$$c = \gamma_1 - \gamma_0 \tag{5}$$

$$\nu = -\tau_0 \times c \times c \tag{6}$$

$$\gamma_m = \frac{1}{2} ||\tau_0|| \, ||c|| \hat{\nu} \tan \frac{\arccos(\max(\frac{\tau_0}{||\tau_0||} \cdot \frac{c}{||c||}, 0)}{2} \tag{7}$$

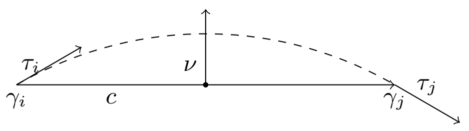

Figure 3: *When splitting a vortex filament with vertices γ_i and γ_j, we place the midpoint as if on an arc with tangents τ_i and τ_i at the vertices. Here, the midpoint will be placed along the ray ν between the line segment c (connecting the vertices) and the arc itself, scaled by the magnitudes of the tangents, which reflects the level of agreement between all filaments attached to each vertex.*

. We cap the size of the arc at half of a full circle, in order to prevent the midpoint from being located more than half the length of the filament segment away from its physical center (since we are splitting the filament, this will be half of its radius), and we further scale the offset between the physical filament center and the center of this arc by the magnitude of the tangent. The procedure is analogous for the second endpoint, except that the tangent and filament directions must be negated.

The actual splitting operation replaces the original filament with two new filaments of half the original strength, connected from the first vertex through the newly created midpoint vertex to the second vertex.

4.1.2 Filament Reconnection

We perform vortex filament reconnection by merging nearby vertices in the vorticity graph, and adjusting the filaments attached to these vertices accordingly. In order to help preserve realism, we have developed several additional constraints controlling when reconnection occurs, beyond simply requiring the vertices to be within half a radius of each other. When determining whether or not to join two vertices, we also consider their relative tangents and the angle between their tangents and their relative offset.

The first observation behind these criteria is that, while reconnections occurring along a filament cause relatively little change in overall velocity, reconnections between multiple filaments, especially when these are two opposing filaments running parallel to each other, can be much more visible. In order to prevent this, we multiply the maximum distance at which the vertices γ_i, and γ_j can be merged by the factor,

$$||\tau_i||\,||\tau_j|| \sqrt{\left|\frac{(\hat{\tau_i}\cdot(\gamma_j-\gamma_i))(\hat{\tau_j}\cdot(\gamma_j-\gamma_i))}{||\gamma_j-\gamma_i||^2}\right|} + (1-||\tau_i||\,||\tau_j||) \quad (8)$$

. This reduces the maximum radius for merges when the offset between the vertices under consideration is perpendicular to either of their tangents, weighted by the magnitude of their tangents.

We also directly enforce a maximum angle between any two filaments being joined, so that passing perpendicular filaments do not immediately merge. This angle is computed as follows,

$$\arccos|\tau_i\cdot\tau_j|\,||\tau_i||\,||\tau_j|| \quad (9)$$

. Once again, we modulate the angle by the magnitudes of the vertex tangents, so that vertices without a predominant vorticity direction are not arbitrarily prevented from reconnecting.

We compute the merged vertex location by taking an average of the positions of the two original vertices, weighted by their aggregate strength. If there is no vortex filament between the vertices to be merged, reconnection is straightforward. All filaments which were previously connected to the original vertex become connected to the new one, and any filaments which become coincident due to the merge are combined by adding their strengths. If there was a filament between the two vertices being merged, we must first redistribute its vorticity to the surrounding filaments. We redistribute the strength of the filament to be eliminated to all filaments which will be attached to the merged vertex, weighted by the dot product between the eliminated filament's direction and the remaining filaments' new directions.

Both this procedure and the the merging that may occur between two filaments which become coincident due to a reconnection may lead to the loss of some vorticity. Though this prevents the simulation from being perfectly energy preserving, we found the total amount of lost energy to be negligible, and the effect is consistent with a fluid of low, but nonzero viscosity. In the simulation shown in figure 2, approximately 20% of the vortex energy is lost over the course of the 1,800 timestep, 30 second simulation.

The reconnection phase is sufficiently robust that in all simulations (except that shown in figure 2, for comparison purposes), we seed low strength vortex rings at every timestep. They simply merge instantly, producing a smaller number of high strength rings leaving the source.

4.2 Smoke Sheets

At its core, our smoke representation simply consists of an indexed triangle mesh, with smoke density values stored at each vertex. The mesh is seeded from an emitter, which is initially constructed as a ring of vertex pairs, one locked in place, and one freely moving, stitched together in a cylindrical configuration (figure 4). As the freely moving vertices are advected by the smoke, our system will begin splitting triangle edges in order to prevent their lengths from exceeding a predefined limit. This causes the smoke to be drawn out of the emitter as a single sheet of triangles joined at common vertices (figure 4).

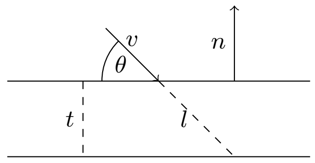

Figure 5: *We treat each face in the smoke sheet as a thin layer with thickness t approaching zero. As we already know the absorption of the sheet when viewed from the normal direction n, all we need is the ratio $\frac{l}{t} = |\csc\theta| = \frac{1}{|n\cdot v|}$.*

As with the vortex filaments, we also occasionally merge vertices in the smoke mesh in order to prevent the vertex density in an area of constriction, or where multiple smoke sheets fold over each other, from becoming unnecessarily high. However, as any change to the smoke mesh is immediately visible, we place additional restrictions on when this may take place.

4.2.1 Density and Rendering

We store smoke density values at each vertex, representing the total absorbency for all the smoke to be drawn due to the vertex. Since the density is distributed over the area of all triangles connected to each vertex, the absorbance at each point within these triangles will decrease as the area of the triangles attached to a vertex increases. We define a density per unit area value for each vertex as follows:

$$\frac{3d}{\sum a_i} \quad (10)$$

, where d is the total density stored at the vertex, and a_i is the area of each face.

Since we are rendering an arbitrary collection of triangles, each vertex does not necessarily have a well defined surface normal. Instead, we adjust for the angle of incidence between the viewing direction and the normal of each triangle independently, and use the sum of their contributions to determine an absorbance value at each vertex. We then interpolate these absorbance values between vertices during rendering.

We will first consider the case of a single triangle with absorbance α per unit area and normal n, viewed from the direction v. We assume that the smoke density is distributed within a thin layer which, when viewed head-on, will have absorption α. Stated differently, this layer has an absorbance of $\frac{\alpha}{t}l$ for a ray which travels l distance through the layer, and a thickness of t (figure 5). Thus, when viewed from an angle θ, the absorbance will be $\frac{\alpha}{t}t\sec\theta$, or simply

$$\alpha\sec\theta = \frac{\alpha}{|n\cdot v|} \quad (11)$$

.

Thus, by distributing the density from a single vertex over its connected faces and summing the absorption for each face from direction v, we obtain the following formula for the total absorption at the vertex,

$$\frac{3d}{\sum a_i|n_i\cdot v|} \quad (12)$$

. For the final rendering step, we interpolate this absorption value across each triangle using an OpenGL shader, and compute the proportion of background light to be absorbed, the alpha value, of each pixel as follows,

$$\exp(-\alpha) \quad (13)$$

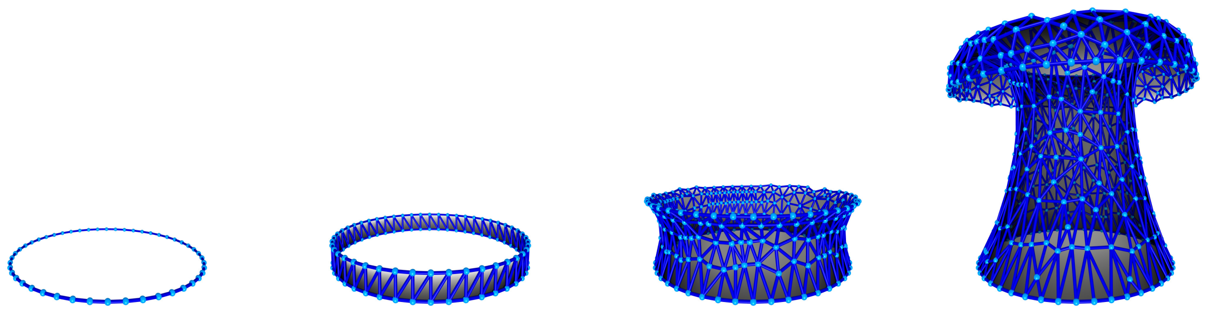

Figure 4: *The smoke source begins as a pair of coincident rings of vertices, one fixed and one advected with the fluid, which are connected to form a cylindrical mesh as they are drawn apart. Once the initial vertices become far enough apart, our triangle splitting procedure begins to insert additional vertices into the mesh, allowing a sheet of smoke to form.*

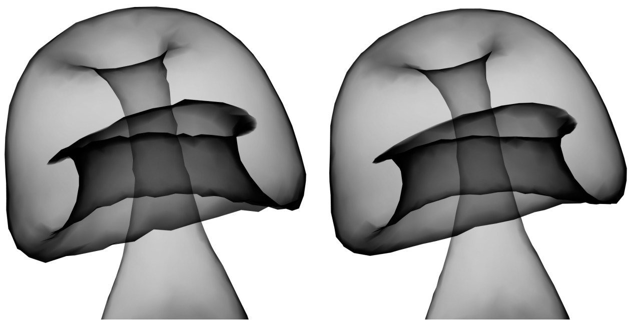

Figure 6: *We remove sharp angles due to our mesh discretization (**left**) using a Laplacian smoothing technique (**right**).*

, where α is the interpolated absorbance from the viewing direction at the pixel. Since we are rendering non-reflective black smoke, the triangle ordering is unimportant.

In order to improve the appearance of sheet boundaries and folds, we also optionally apply laplacian smoothing to the vertices in the smoke mesh before rendering each frame (figure 6). This is done by using the weighted average of neighbor vertex location in the place of each vertex location. We weight by the inverse of the vertex coverage, as we found that this produced smoother edges. Without this weighting, vertices on an edge or fold with more internal neighbors may be pulled further toward the interior of the mesh than other nearby vertices, negating the overall smoothing effect.

4.2.2 Triangle Splitting

Like vortex filaments, triangle edges are split whenever any of their edge lengths exceeds a user-defined threshold, S_d. However, we also split triangle edges when the angle between the velocities at each vertex exceeds a second user-defined threshold, S_θ. This is to ensure that the triangle mesh can properly track the natural smoke curvature as it curls around a vortex filament. On the rare 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 total, we select an edge for splitting if either of the following conditions are met:

$$||\gamma_j - \gamma_i|| > S_d \tag{14}$$

$$\frac{u(\gamma_i) \cdot u(\gamma_j)}{||u(\gamma_i)||\,||u(\gamma_j)||} < \cos S_\theta \quad \text{and} \quad ||\gamma_j - \gamma_i|| > S_m \tag{15}$$

, where γ_i is the position of vertex i and $u(\gamma_i)$ is the velocity at it's location.

To split the edge, we simply insert a new vertex γ_m and split each triangle containing the vertices γ_i and γ_j into one triangle containing γ_i and γ_m, and a second containing γ_m and γ_j. We then place the new vertex at the average of the positions of all vertices it is connected to, weighted by the inverse of their coverage.

Our concept of vertex *coverage* is derived from the idea that a vertex in an approximately flat sheet of geometry will normally be surrounded on all sides by triangles which are located near the tangent plane of that vertex. Thus, the sum of all angles which the vertex is part of will be approximately 360°. Our coverage value is simply the sum of all angles which a vertex is part of, divided by 360°. It provides a rough estimate of the configuration of local geometry surrounding the vertex. If the vertex is on the edge of a sheet, it's coverage will be near $\frac{1}{2}$. If it is in a single sheet, the coverage will be near 1, and if it is part of a more complex configuration, the coverage may be higher.

By using the coverage to help determine the splitting vertex location, we can place it near locations in which additional geometric information is more useful, such as near the edge of a sheet. This also allows us to place vertices based on the location of all surrounding vertices without significantly eroding sheet edges. It would be possible to prevent erosion by simply placing splitting vertices at the center of the edge they are splitting. However, this would also cause unnatural creeping and folding when splitting curved regions, as the splitting vertex could be located significantly closer to the local center of curvature than surrounding vertices.

The final step in triangle splitting is to redistribute vertex density. This procedure is described in detail in section 4.2.4, as it is common to both splitting and joining. 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 nearly always reduce the area of triangles connected to 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.

4.2.3 Triangle Reconnection

As with filament reconnection, we perform triangle reconnection by merging adjacent vertices in the smoke mesh. Once again, the basic requirement is that the distance between the vertices is at most half of the splitting distance, and we have developed several additional constraints to help preserve mesh detail under certain circumstances.

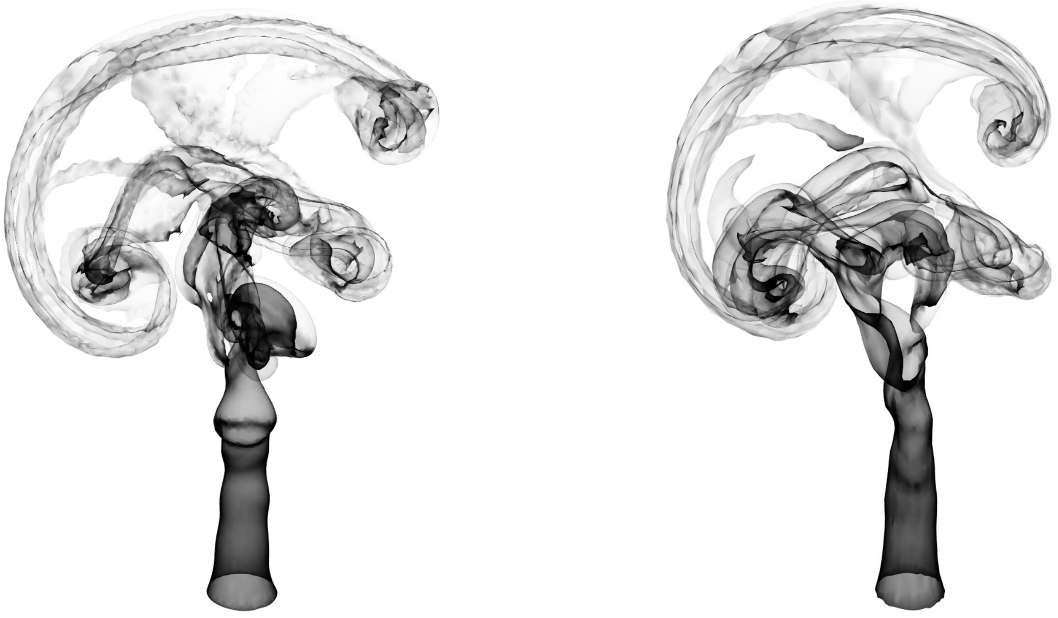

Figure 7: *An adaptive splitting and reconnection distance (**right**) generates significantly smoother smoke and preserves fine details better than than a constant distance (**left**), despite using just over $\frac{2}{3}$ the number of total triangles.*

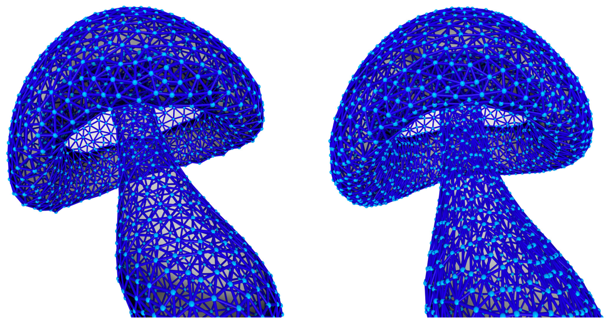

Figure 8: *Without the additional constraints in equations 15, 16, and 17, detail is lost in the constricting column in the center, as well as the curl under the plume (**left**). With these additions, detail is preserved in these areas at the expense of vertex density in flatter areas of the plume (**right**). (The splitting distance used in the left simulation is slightly longer in order to achieve similar total vertex counts.)*

We again multiply the maximum reconnection distance by a factor, guaranteed to be less than 1, which is designed to help preserve edge detail and locally increase the density of vertices in directions perpendicular to the fluid flow. This factor is more easily discussed as two separate components.

The first, responsible for preserving edges, is simply the ratio of coverage between the vertex with lesser coverage and the vertex with greater coverage. When both vertices have similar coverage, for instance if they are both part of a flat sheet-like region, then this factor will have no effect. However, if one vertex is part of the border of a sheet and the other is not, for instance, the maximum reconnection distance will be roughly cut in half.

The second factor is similar to the factor we apply to the filament reconnection distance. It is designed to allow reconnection freely along the direction of the velocity field, but reduce the reconnection distance perpendicular to it:

$$\sqrt{\left|\frac{(u(\gamma_i)\cdot(\gamma_j-\gamma_i))(u(\gamma_j)\cdot(\gamma_j-\gamma_i))}{||u(\gamma_i)||\,||u(\gamma_j)||\,||\gamma_j-\gamma_i||^2}\right|} \tag{16}$$

(figure 8).

Finally, we add a second constraint to maintain consistency with that expressed in equation 15:

$$\frac{u(\gamma_i)\cdot u(\gamma_j)}{||u(\gamma_i)||\,||u(\gamma_j)||} > \cos\frac{1}{2}S_\theta \quad \text{and} \quad |\gamma_j-\gamma_i| < \frac{1}{2}S_m \tag{17}$$

. In general, while a reconnection constraint need not be mirrored in the splitting constraints, all splitting constraints must be mirrored with similar reconnection constraints, so that a vertex is not reconnected immediately after being split.

Once a pair of vertices is selected for reconnection, we begin by computing the location of the new, reconnected vertex. We do this by averaging the old locations, weighted by their respective ratios of areas to coverage. This will tend to place the new vertex such that it is surrounded by a roughly even 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 effected triangles due to the reconnection. As significant changes in surface normal may produce significant changes in effective absorption from the viewing direction, we do not perform the reconnection if the angle between the old and new surface normals of any triangles are beyond a user-defined threshold. This does reduce reconnections between parallel sheets of smoke, but also noticeably reduces popping artifacts due to surface normal changes.

If this test passes, the reconnection is performed by removing any triangles which contain both old vertices, as they would become degenerate, and swapping each old vertex for the new reconnected vertex in the remaining triangles. As with splitting, we must now redistribute the vertex densities, this time seeding the pool of extra density with the density from the two vertices being joined.

Overall, we found that using a varying mesh density improved results while allowing the use of a slightly larger maximum triangle splitting distance. In figure 7, the simulation with a dynamic splitting distance yields noticeably better results despite using 29% fewer triangles.

4.2.4 Density Redistribution

When we split or reconnect triangles, we must redistribute the density among the vertices adjacent to the modified vertex in order to maintain consistent absorbance values, or density per unit area. In each case, we first attempt to maintain the ratio of density to area in all adjacent vertices, and place the remaining density in the new vertex.

We begin by storing the ratio of density to area in each vertex prior to the splitting. We adjust the density in each neighbor vertex to match the old ratio, and keep track of the total difference in density. In the case of splitting, this will usually leave excess density, but in the case of reconnection, this will usually require extra density. With reconnections, however, we also add all the density from the two merged vertices to this pool.

If the value of the excess density pool is positive, we simply store it in the new vertex. If there is a shortage of density, we store no density in the new vertex, and remove density from all other vertices proportional to their current total density.

We distribute density in this manner in order to 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. 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.

The one constant between these cases is that all vertices not being merged 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 Results

Our vortex filament reconnection method is able to preserve accurate simulation results while significantly reducing the number of simulated filaments. In figure 2, we compare the results of otherwise identical simulations with and without reconnection (reconnection is still allowed between adjacent vertices in both cases). Without reconnection, the 1,800 frame simulation completes in 851 seconds, and results in 4,866 filaments and vertices. With reconnection, the simulation takes only 102 seconds (just under $\frac{1}{3}$ of real time), and results in only 1518 filaments and 1391 vertices.

To enable a fair comparison, in the simulation of figure 2, we spawn one filament ring every 0.6 seconds, so that the number of rings leaving the jet is identical both with and without reconnection. However, in all other simulation, we spawn a weaker ring at every timestep, and allow our reconnection technique to merge it with the nearby ring from previous timesteps. Since we weight the reconnected vertex location by strength, the ring will eventually leave the source despite the reconnections, and a new one will begin forming.

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). We achieve results of similar quality to those demonstrated by Weißmann and Pinkall [2010] while using less than $\frac{1}{10}$ of the tracking locations.

Our results in figure 7 demonstrate the effectiveness of our triangle splitting and reconnection criteria. Despite having significantly more artifacts in regions of sheet reconnection, and losing detail in the central smoke column where it becomes constricted, the left-hand simulation with a fixed splitting and reconnection radius uses 29,449 smoke vertices, while the simulation with our criteria uses only 21,008. (We increase the maximum reconnection range in the second simulation, as otherwise it would inevitably end up with more vertices.)

The additional complexity of our dynamic vertex density criteria do, however, add some computational overhead, requiring 876 seconds to complete the 900 frame animation, as opposed to 631 with a constant density. This is most likely due to the need to expand the search for reconnections beyond the closest available vertex, since the closest vertex may fail our additional criteria while another more distant one that still falls within the maximum reconnection range will pass.

6 Discussion

In this paper, we introduce both an extension to existing vortex filament based fluid simulation methods, as well as a smoke tracking and rendering system designed to minimize the number of points to be advected, and thus the number of velocity computations to be performed. Our simulation is able to retain a high degree of physical accuracy while significantly reducing the total number of filaments through reconnection, and our smoke representation and advection method is able to preserve thin sheet-like formations without dissipation using far fewer points of advection than would be required to achieve a similar result with a purely particle-based method.

Though our smoke representation preserves details without significant dissipation, an area of future work might be to explore adding some degree of simulated dissipation to the final rendered result. Indeed, few rendering techniques for smoke take as direct an approach as our own, which simply draws the smoke mesh, and instead perform a significant amount of post processing to produce the final result. This might give the smoke spawned early in a simulation a more natural appearance, and allow sheets which have become close enough to begin reconnecting to smoothly blend into each other as they merge.

Beyond this, it would certainly be worthwhile to explore volumetric rendering of our smoke mesh. By allowing each triangle to define a smoothed region of smoke, whose shape is controlled by the local mesh geometry, we might extend this technique to support incorporation with existing photorealistic rendering systems including support of global illumination models.

Regarding our vortex reconnection model, though we have shown that a visually inspired approach to reconnection can lead to plausible results, we might extend this further by minimizing the total change in velocity due to each reconnection. While still allowing for arbitrary reconnections, one might be able to construct a constraint that finds all edge pairs which can be reconnected with a net velocity change less that some maximum, and then proceeds to place the reconnected vertex such that it minimizes this change. Splitting vertices accurately, however, would remain more prob-

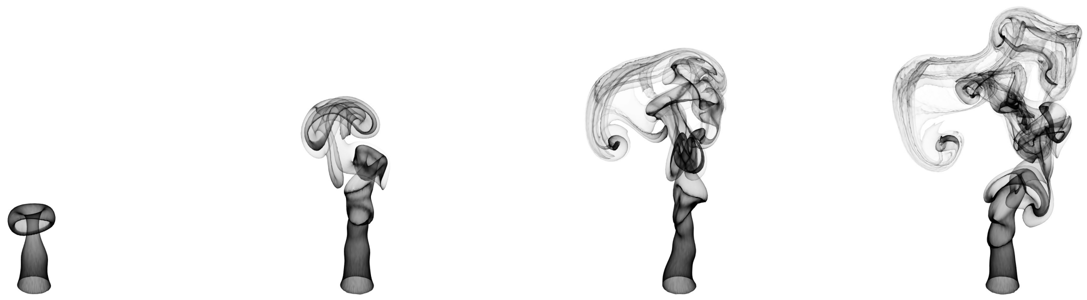

Figure 9: *The progression of a high resolution simulated smoke surface. The last frame of this sequences uses 92830 vertices in the smoke mesh.*

lematic, as an velocity change minimizing approach would simply preserve the original straight shape.

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. These techniques may have the greatest potential in real-time applications, where low complexity is more important than photo realism, as each degrades well in quality with reduced complexity. The greatest advantage of each of these methods is that computational complexity depends directly on the number of features currently being simulated, and not the total capacity of the simulation environment.

References

ANGELIDIS, A., AND NEYRET, F. 2005. Simulation of smoke based on vortex filament primitives. In *Symposium on Computer Animation, SCA 2005, July, 2005*, ACM Press, Los Angeles, California, Etats-Unis, ACM-SIGGRAPH/EG, 87–96.

ANGELIDIS, A., NEYRET, F., SINGH, K., AND NOWROUZEZAHRAI, D. 2006. A controllable, fast and stable basis for vortex based smoke simulation. In *Symposium on Computer Animation, SCA 06, September, 2006*, Eurographics, Vienne, Autriche, C. O'sullivan and F. H. Pighin, Eds., ACM-SIGGRAPH/EG, 25–32.

CHATELAIN, P., CURIONI, A., BERGDORF, M., ROSSINELLI, D., ANDREONI, W., AND KOUMOUTSAKOS, P. 2008. Billion vortex particle direct numerical simulations of aircraft wakes. *Computer Methods in Applied Mechanics and Engineering 197*, 13-16, 1296 – 1304.

CUMMINS, S. J., AND RUDMAN, M. 1999. An sph projection method. *Journal of Computational Physics 152*, 2, 584 – 607.

DESBRUN, M., AND PAULE GASCUEL, M. 1996. 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*, Springer-Verlag, 61–76.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26* (January).

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

KRGER, J., AND WESTERMANN, R. 2005. Gpu simulation and rendering of volumetric effects for computer games and virtual environments. In *IN PROCEEDINGS*, Eurographics, 685–693.

NARAIN, R., SEWALL, J., CARLSON, M., AND LIN, M. C. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph. 27* (December), 166:1–166:8.

PARK, J., SEOL, Y., CORDIER, F., AND NOH, J. 2010. A smoke visualization model for capturing surface-like features. *Computer Graphics Forum 29*, 2352–2362.

PINKALL, U., SPRINGBORN, B., AND WEISSMANN, S. 2007. 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*, 12563–12576.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics 24*, 910–914.

STAM, J., AND FIUME, E. 1995. Depicting fire and other gaseous phenomena using diffusion processes. 129–136.

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

TAN, J., AND YANG, X. 2009. Physically-based fluid animation: A survey. *Science in China Series F-Information Sciences 52*, 5, 723–740.

VON FUNCK, W., WEINKAUF, T., THEISEL, H., AND SEIDEL, H.-P. 2008. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics 14*, 1396–1403.

WEISSMANN, S., AND PINKALL, U. 2009. Real-time interactive simulation of smoke using discrete integrable vortex filaments. In *VRIPHYS*, 1–10.

WEISSMANN, S., AND PINKALL, U. 2010. Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph. 29* (July), 115:1–115:12.