Smoke Sheets and Vortex Filaments with Flexible Reconnection

Alfred Barnat

CMU-CS-11-123

July 2011

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Nancy S. Pollard, Chair
Adrien Treuille

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Copyright © 2011 Alfred Barnat

Keywords: Graphics, Animation, Fluid Simulation, Smoke, Vorticity, Vortex Filaments

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. 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 the entire simulation space. We build upon previous work which merges adjacent rings of filaments by allowing filaments to form structures other than rings and developing a new set of reconnection criteria to take advantage of this generic filament graph. 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.

Acknowledgements

I would like to thank my advisor, Nancy Pollard, for introducing me to graphics research and for her support and guidance throughout this project and those that came before it, as well as Adrien Treuille for always being available to answer my questions and for his advice and feedback throughout this project.

I would also like to thank Peter Steenkiste and Deborah Cavlovich for running this program and for helping me to answer every question and work through every problem that came up as I worked to complete it.

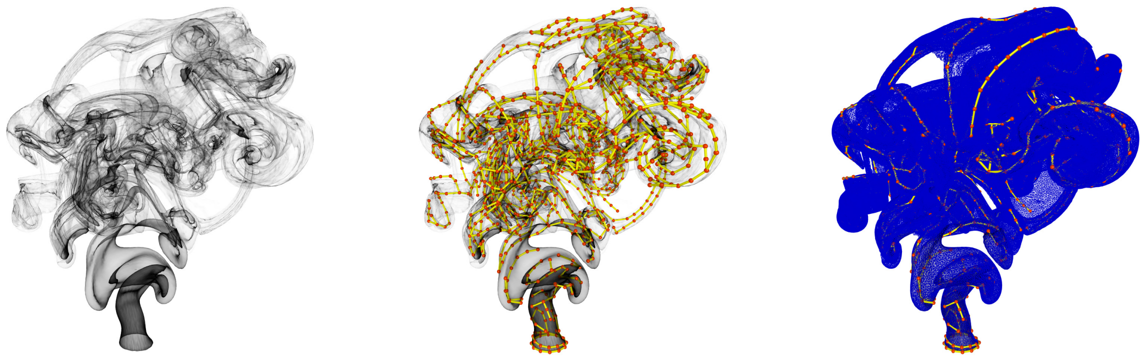

*Figure 1: A smoke plume simulated using our method (**left**). Fluid motion is simulated using vortex filaments (**center**) and the smoke surface is tracked using a triangle mesh (**right**).*

1 Introduction

The majority of smoke simulation methods take an Eulerian approach to the problem [8, 18, 16]. 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 using 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 [17]. 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 [5].

Vortex filaments offer a compromise between these two techniques [2, 13, 20]. Though Lagrangian in nature, they define the fluid velocity through a series of filaments of vorticity rather than through particles of momentum. This eliminates the need for a projection operation, since the vortex filaments define a divergence-free velocity field by construction. 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 [20].

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 convincing animation, since the sheets of vorticity that form as an inviscid 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 [2].

Previous work on vortex filaments has limited itself to ring-structured vorticity. This is a natural simplification, since vortex rings remain stable even in the presence of small amounts of viscosity, and for computational reasons. Specifically, they can be easily simplified through projection into a limited frequency space [2], and they enable the use of more accurate self advection techniques [13].

We extend this work by relaxing the requirement that filaments exist only as a set of distinct, closed loops, and taking advantage of the additional flexibility to support reconnections between any pair of vertices in the filament graph. This allows us to both simplify the reconnection procedure and to support any velocity field with a minimal number of total vortex filaments. 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 that used 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 [16], with the addition of vorticity confinement [8] 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 [18]. 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 [6, 17]. 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 (in which case the air is often assumed to exert no force on the water, and is simply omitted from the simulation).

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 [8, 11, 15]. *Simplical Fluids* [7] 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 [3, 15]. However, similar difficulties in information propagation arise, making them ill-suited for situations in which small timesteps are otherwise unnecessary, such as smoke simulation.

Angelidis and Neyret [1] pioneered the use of vortex filaments as a simulation primitive. Since filaments of vorticity naturally form in a turbulent flow, a sparse basis is sufficient to approximate complex fluid motion. Pinkall et al. [13] further developed a means to accurately model self-advection of discretized polygonal filaments, and Weißmann and Pinkall [20] 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 [8]. 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 [10]. Funck et al. [9] 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 [12]. However, neither of these methods modify particle placements or densities once they have been spawned.

3 Contributions

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 in a directed graph of vortex filaments. This increased flexibility in reconnections allows our simulator to represent complex flows with a minimum number of vortex filaments.

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

4 Physical Motivation

Our vortex filament based simulator allows for the construction of any directed graph of vortex filaments. This representation itself is inspired by previous work, including Simplical Fluids [7], which represents generic fluid motion using vorticity defined on the edges of a space-filling mesh. In regions of our simulation where the graph structure consists of a dense set of vortex edges, the primary difference over short time periods between our technique and Simplical Fluids is that we advect vertices rather than values assigned to each edge.

The strength of vortex filament methods is that filaments need only be defined in limited regions of the fluid. This leads to the primary motivation for vortex reconnection: the interaction of individual pairs of vortex filaments. It is well understood that regions of opposing vorticity will tend to attract each other and merge over time [14, 4]. This principal applies to any pair of nearby vortices, not only those which are already parallel, leading even perpendicular vortices to be drawn toward each other and deflected such that their vorticity aligns in opposing directions.

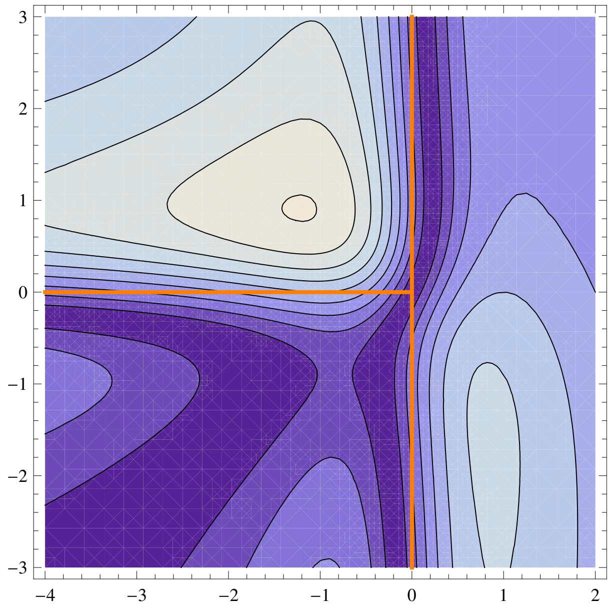

*Figure 2: Cross section of velocity magnitude surrounding perpendicular filament (**orange**) intersection at origin. The centers of rotation, shown by the regions of least velocity (**purple**), bend towards each other in the direction of opposing vorticity as they merge. The filaments each have a radius of 1 and unbounded length in all directions in which they leave the region shown.*

Perpendicular $\perp$ and $+$ shaped junctions which form due to vortex reconnections are an approximation of this effect. 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 filaments, 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.

5 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 some velocity field.

5.1 Filament-based Fluid Simulation

Our fluid simulator is based on the smoothed filament model of Weißmann and Pinkall [19]. Though they suggest the use of doubly discrete smoke ring flow [13] 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,

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

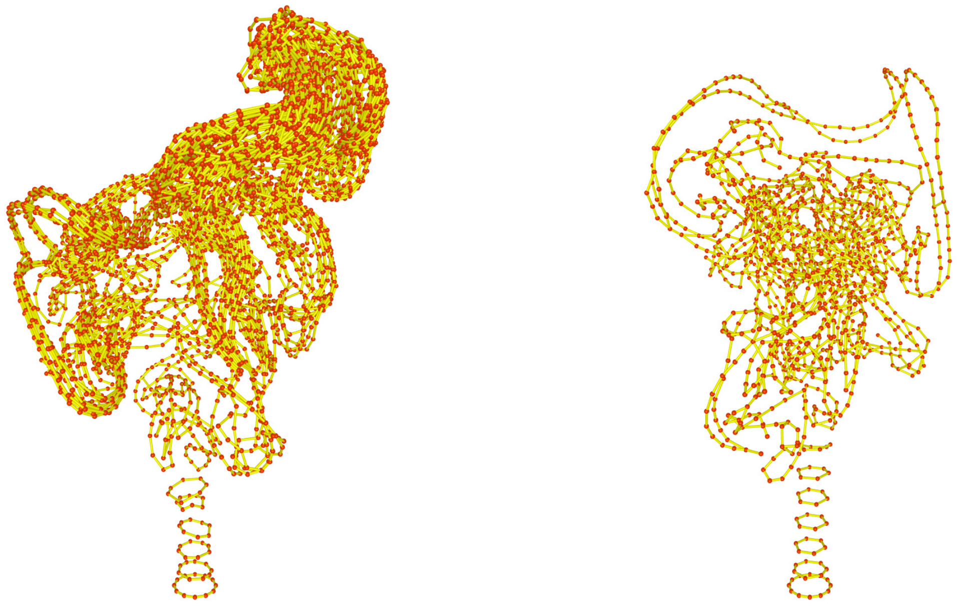

*Figure 3: Without reconnection, thick bundles of vortex filaments form (**left**). With reconnection, the simulation contains $\frac{1}{3}$ the number of filaments, though a small amount of viscosity is introduced (**right**).*

[19]. Here, $u(0)$ is the velocity at the origin due to a filament of strength 4π from vertices γ_i to γ_j, with smoothing radius a. In order to obtain $u(x)$, simply replace γ_i with $\gamma_i - x$.

5.1.1 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. (For reference, the smoothing radius in figure 1 is 0.1, while the view covers a range from -1.2 to 1.2 both horizontally and vertically at the origin.)

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 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 using filament smoothing radii, and aggregate strength values are computed by summing each filament's direction multiplied by its strength and taking the length of the resulting vector.

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

$$a = \frac{\sum a_i \Gamma_i}{\sum \Gamma_i} \tag{3}$$

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

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

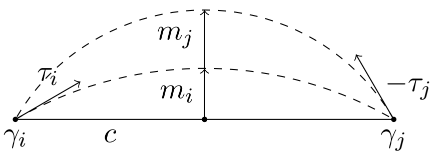

Figure 4: When splitting the vortex filament c with vertices γ_i and γ_j, we compute separate offsets m_i and m_j between the midpoint of the filament and the midpoint of the arc constructed according to the tangents at each vertex, τ_i and τ_j. We then average these offsets and multiply by the product of tangent magnitudes (which represent the agreement at each vertex on the tangent value between the various connected filaments) 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):

$$c = \gamma_j - \gamma_i \tag{5}$$

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

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

, where m is the offset for the given tangent. 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 tangent, 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.

5.1.2 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,

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

. 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:

$$\arccos|\hat{\tau}_i \cdot \hat{\tau}_j|\,||\tau_i||\,||\tau_j|| \tag{9}$$

. 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 do this by redistributing 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 3, 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 3, 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.

5.1.3 Filament Dissipation

As a final step to reduce the number of filaments in our simulation, we cause the vorticity of a filament to dissipate into neighboring filaments (which share a vertex with the filament in question) whenever its strength is less than the average strength of the neighboring filaments by some user-defined constant. The dissipation is performed in a manner analogous to the dissipation of filaments which are removed due to reconnection. The strength is distributed among neighboring filaments, weighted by the dot product between the direction of the filament being dissipated and the direction of the neighboring filament.

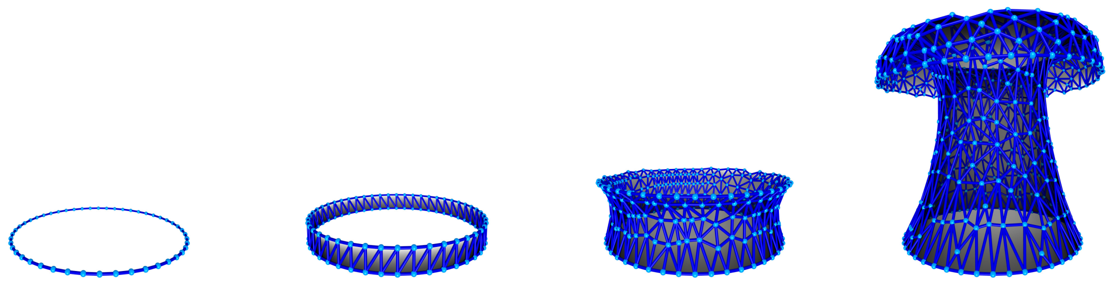

Figure 5: The smoke source begins as a pair of coincident rings of vertices, one fixed and one advected with the fluid, which are connected to form a cylindrical mesh as they are drawn apart. Once the initial vertices become far enough apart, our triangle splitting procedure begins to insert additional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

5.2 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. 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 5).

As with the vortex filaments, we also merge vertices in the smoke mesh in order to prevent the vertex density in areas of constriction and areas 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, which we describe below.

5.2.1 Density and Rendering

We store smoke density values at each vertex, representing the total absorbency for all the smoke to be rendered 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:

$$\frac{3d}{\sum a_i} \tag{10}$$

, 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

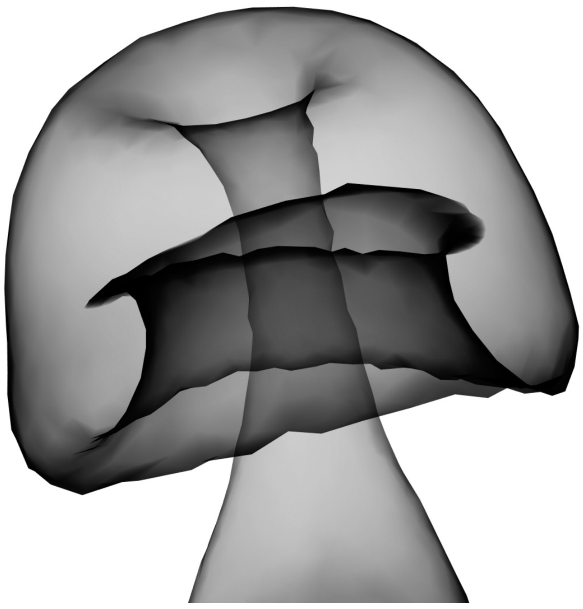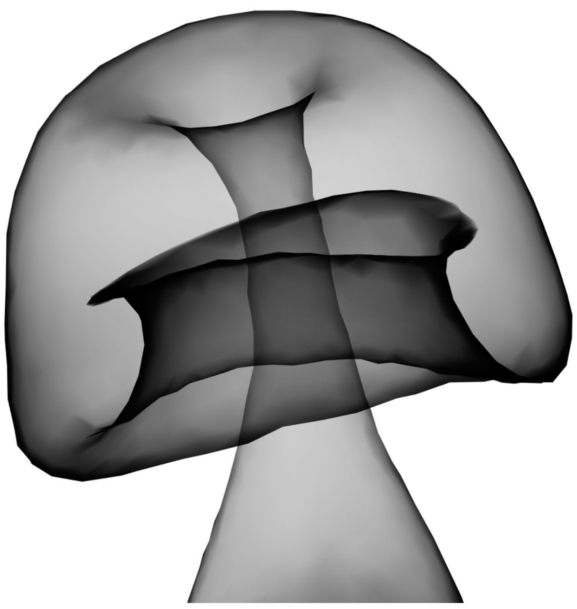

*Figure 6: We remove sharp angles due to our mesh discretization (**left**) using a Laplacian smoothing technique (**right**).*

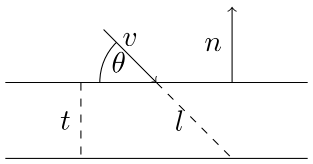

Figure 7: We treat each face in the smoke sheet as a thin layer with thickness t approaching zero. As we already know the absorption of the sheet when viewed from the normal direction n, all we need is the ratio $\frac{l}{t} = |\csc\theta| = \frac{1}{|n\cdot v|}$.

absorbance of $\frac{\alpha}{t}l$ for a ray which travels l distance through the layer, and a thickness of t (figure 7). Thus, when viewed from an angle θ, the absorbance will be $\frac{\alpha}{t}t\sec\theta$, or simply

$$\alpha\sec\theta = \frac{\alpha}{|n\cdot v|} \tag{11}$$

.

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:

$$\frac{3d}{\sum a_i|n_i\cdot v|} \tag{12}$$

. 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,

$$\exp(-\alpha) \tag{13}$$

, 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.

5.2.2 Triangle Splitting

Like vortex filaments, triangles are split whenever any of their edge lengths exceed 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. Overall, 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, its coverage will be near $\frac{1}{2}$. If it is in the middle of 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. (Vertices located near the center of an edge being split will track the curvature of the velocity field, but the edge itself simply represents the shortest path between it's endpoints.)

The final step in triangle splitting is to redistribute vertex density. This procedure is described in detail in section 5.2.4, as it is common to both splitting and reconnection. Briefly, we attempt to

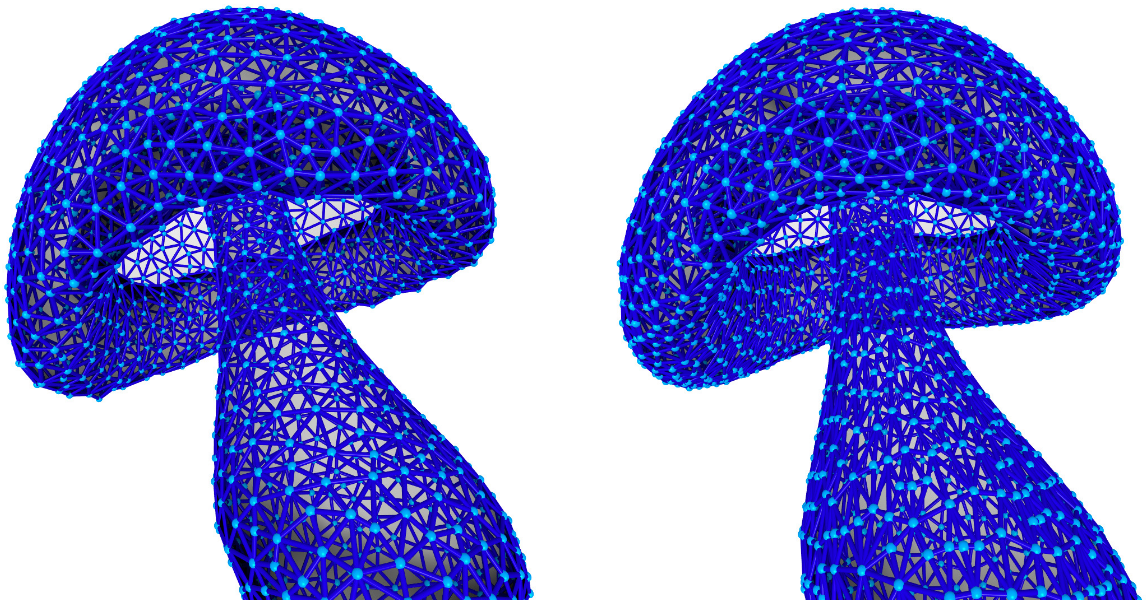

*Figure 8: Without the additional constraints in equations 15, 16, and 17, detail is lost in the constricting column in the center, as well as the curl under the plume (**left**). With these additions, detail is preserved in these areas at the expense of vertex density in flatter areas of the plume (**right**).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

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.

5.2.3 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.

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 (e.g. 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, 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

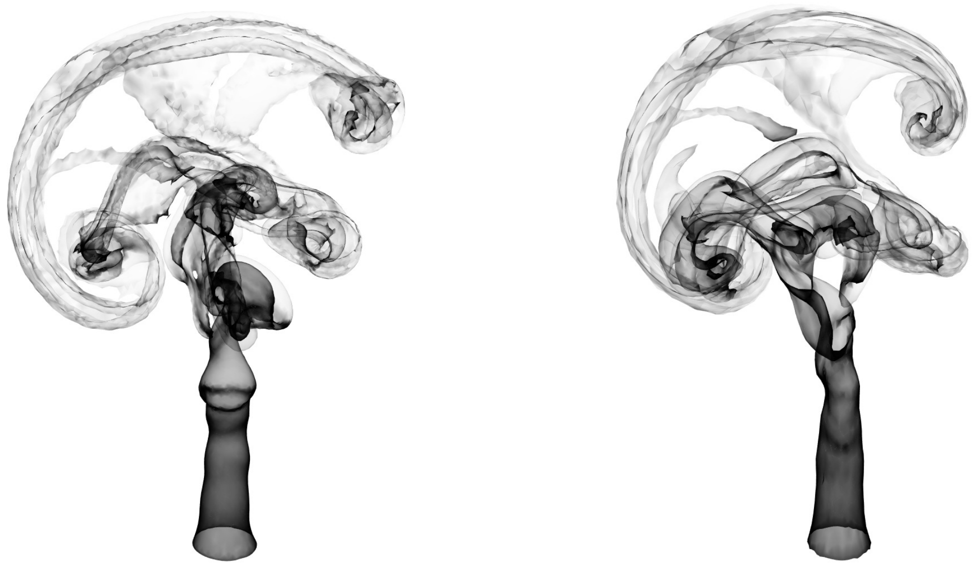
*Figure 9: An adaptive splitting and reconnection distance (**right**) generates significantly smoother smoke and preserves fine details better than than a constant distance (**left**), despite using just over $\frac{2}{3}$ the number of total triangles.*

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 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 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 changes in surface normals.

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 9, the simulation with a dynamic splitting distance yields visibly better results despite using 29% fewer triangles.

5.2.4 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. This method of redistribution preserves total simulation density by construction.

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.

In both of these cases, 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.

6 Results

Our vortex filament reconnection method is able to produce physically plausible simulation results while significantly reducing the number of simulated filaments. In figure 3, 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 1,518 filaments and 1,391 vertices. This is a reduction to about 9% of the original simulation complexity (measured by the total number of individual edge velocity computations required, which is simply the product of the number of edges and the number of vertices), as compared with a reduction to 12% of the original complexity achieved by the reconnection method of Weißmann and Pinkall [20].

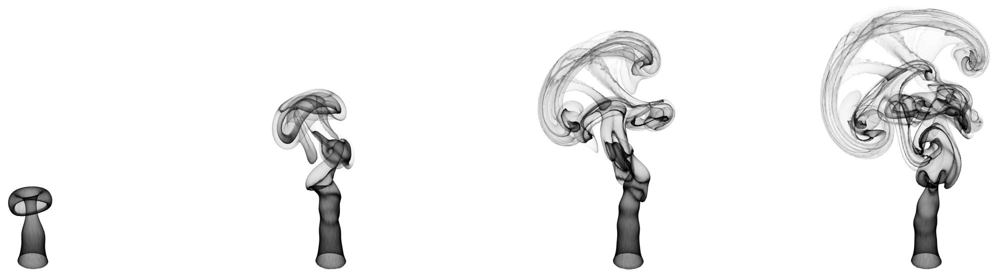

Figure 10: The progression of a high resolution simulated smoke surface. In the rightmost frame, the smoke mesh contains 76,077 vertices.

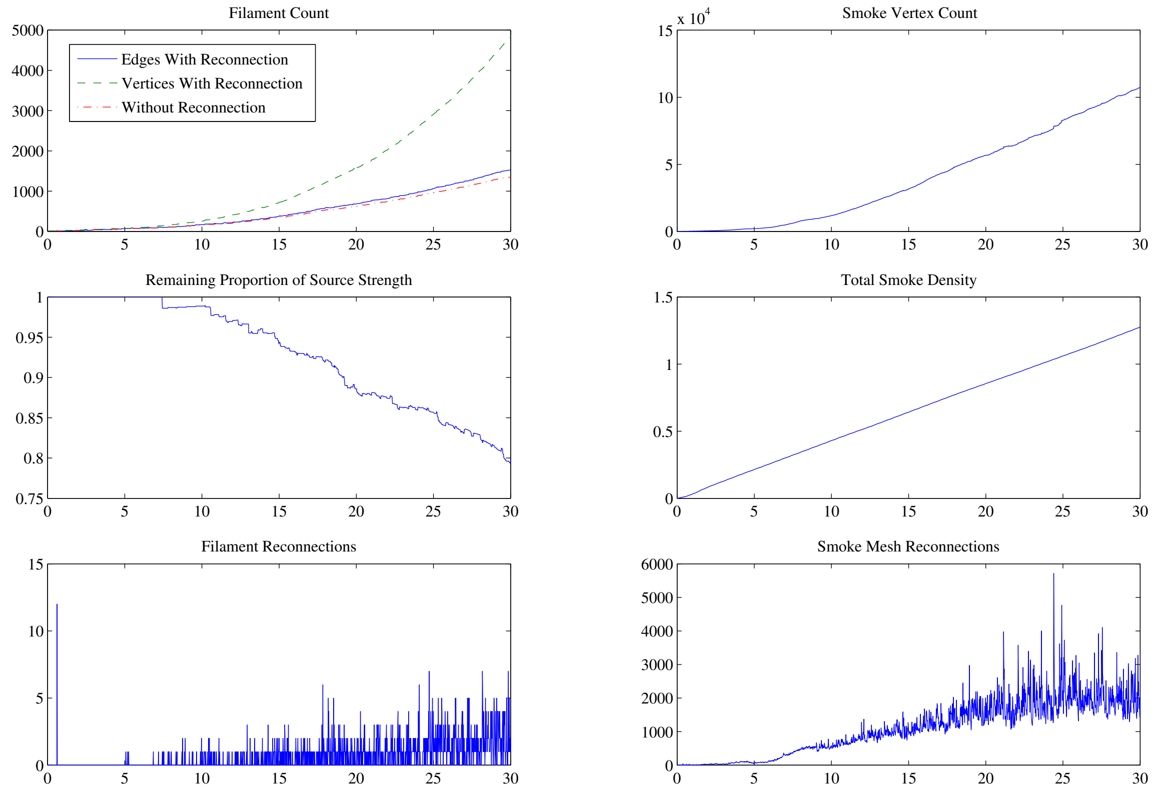

Figure 11: Trends of a 30 second, 1,800 frame simulation.

To enable a fair comparison, in the simulation of figure 3, 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 [20] while using less than $\frac{1}{10}$ of the tracking locations (figure 10).

Our results in figure 9 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 right-hand simulation with our modified 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.

In figure 11, we plot several statistics of a simulation over time. Note that both the filament count with reconnection and smoke vertex count curves flatten as the simulation progresses. Since turbulence that forms ahead of the jet causes large-scale fluid motion to quickly dissipate, both filaments and smoke become trapped in the region immediately in front of the source. As the volume of this region becomes filled, reconnection prevents the smoke and filament vertex densities from increasing beyond their respective thresholds. This effect is also visible in the reconnection rates, which both rise roughly in proportion to the total number of vertices.

The total smoke density plot demonstrates the correctness of our density redistribution step. As the velocity at the source is near constant, so is the rate of density introduction into the simulation. Filament strength is also introduced at a constant rate. However, due to cancellation during reconnections, it is also lost at a roughly constant rate.

7 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 realism 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 we have shown that a visually inspired approach to reconnection can lead to plausible results, we might extend this further by attempting to minimize the total change in velocity due to each reconnection. While still allowing for arbitrary reconnections, one might construct a constraint that finds all edge pairs which can be reconnected with a net velocity change less that some maximum value, and then proceeds to place the reconnected vertex in order to minimizes this change.

Though our filament graph structure does not necessitate a change to previous methods of computing boundary vortices, smoke must also be handled in such a way that it does not intersect with object boundaries. This may necessitate the introduction of intersection tests between smoke triangles and object surfaces, in order to split triangles to conform to the surface.

Smoke smoothing may offer another solution to this problem, as well as a way to reduce artifacts due to reconnections. As our smoke mesh retains surface orientations within the smoke, it may be possible to introduce a smoothing factor along the plane of the smoke. This would reduce the visibility of boundaries at sheet edges, and smooth some of the triangular artifacts that occur during reconnection.

This technique might be further extended to introduce simulated diffusion to our non-diffuse surfaces. Though low diffusion is desirable for smoke renderings, the complete lack of diffusion generated by our method can appear somewhat unrealistic over long simulation periods. By rendering mesh triangles as a diffuse volume of increasing thickness, we might reintroduce a small amount of diffusion while simultaneously allowing existing volumetric smoke rendering techniques to be directly applied to our mesh.

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

[1] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 87–96, New York, NY, USA, 2005. ACM. ISBN 1-59593-198-8. doi: 10.1145/1073368.1073380.

[2] Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. A controllable, fast and stable basis for vortex based smoke simulation, 2006. URL `http://portal.acm.org/citation.cfm?id=1218064.1218068`.

[3] Philippe Chatelain, Alessandro Curioni, Michael Bergdorf, Diego Rossinelli, Wanda Andreoni, and Petros Koumoutsakos. Billion vortex particle direct numerical simulations of aircraft wakes. *Computer Methods in Applied Mechanics and Engineering*, 197(13-16):1296 – 1304, 2008. ISSN 0045-7825. doi: 10.1016/j.cma.2007.11.016. URL `http://www.sciencedirect.com/science/article/pii/S0045782507004574`.

[4] Alexandre Joel Chorin. Hairpin removal in vortex interactions ii. *J. Comput. Phys.*, 107:1–9, July 1993. ISSN 0021-9991. doi: 10.1006/jcph.1993.1120.

[5] Sharen J. Cummins and Murray Rudman. An sph projection method. *Journal of Computational Physics*, 152(2):584 – 607, 1999. ISSN 0021-9991. doi: 10.1006/jcph.1999.6246. URL `http://www.sciencedirect.com/science/article/pii/S0021999199962460`.

[6] Mathieu Desbrun and Marie-paule Gascuel. 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*, pages 61–76. Springer-Verlag, 1996.

[7] Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.*, 26, January 2007. ISSN 0730-0301. doi: 10.1145/1189762.1189766.

[8] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 15–22, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. doi: 10.1145/383259.383260.

[9] Wolfram von Funck, Tino Weinkauf, Holger Theisel, and Hans-Peter Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics*, 14:1396–1403, 2008. ISSN 1077-2626. doi: 10.1109/TVCG.2008.163.

[10] Jens Krüger and Rüdiger Westermann. Gpu simulation and rendering of volumetric effects for computer games and virtual environments. *Computer Graphics Forum*, 24(3):685–693, 2005. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2005.00893.x.

[11] Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Trans. Graph.*, 27:166:1–166:8, December 2008. ISSN 0730-0301. doi: 10.1145/1409060.1409119.

[12] Jinho Park, Yeongho Seol, Frederic Cordier, and Junyong Noh. A smoke visualization model for capturing surface-like features. *Computer Graphics Forum*, 29:2352–2362, 2010. doi: 10.1111/j.1467-8659.2010.01719.x.

[13] Ulrich Pinkall, Boris Springborn, and Steffen Weißmann. 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, 2007. doi: 10.1088/1751-8113/40/42/S04.

[14] P. G. Saffman. A model of vortex reconnection. *Journal of Fluid Mechanics*, 212:395–402, 1990.

[15] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics*, 24:910–914, 2005. doi: 10.1145/1073204.1073282.

[16] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: 10.1145/311535.311548.

[17] Jos Stam and Fiu Eugene. Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 129–136, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: doi.acm.org/10.1145/218380.218430.

[18] Jie Tan and XuBo Yang. Physically-based fluid animation: A survey. *Science in China Series F-Information Sciences*, 52(5):723–740, 2009. doi: 10.1007/s11432-009-0091-z.

[19] Steffen Weißmann and Ulrich Pinkall. Real-time interactive simulation of smoke using discrete integrable vortex filaments. In *VRIPHYS*, pages 1–10, 2009. doi: 10.2312/PE/vriphys/vriphys09/001-010.

[20] Steffen Weißmann and Ulrich Pinkall. Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph.*, 29:115:1–115:12, July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1778852.