Particle Flurries

Synoptic 3D Pulsatile Flow Visualization

Jason S. Sobel, Andrew S. Forsberg, David H. Laidlaw, Robert C. Zeleznik, Daniel F. Keefe, Igor Pivkin, George E. Karniadakis, Peter Richardson, and Sharon Swartz
Brown University

Particle Flurries is an interactive approach to 3D flow visualization. The approach produces a "synoptic visualization" and is used to examine both internal and external flows.

Synoptic visualizations give viewers a synopsis of all flow features simultaneously. Good examples of 2D synoptic visualizations are weather maps from both TV news reports and online Web pages. The human visual system is adept at finding patterns within larger contexts, and we hypothesize that synoptic visualization methods will help users find unexpected features more quickly and thus speed the understanding of complex 3D time-varying flows.

Particle Flurries (PF) is our effort toward a synoptic visualization of complex pulsatile 3D flow. Our group's ongoing study of the correlation between arterial blood flow and lesions as well as our research into the mechanics, dynamics, and evolution of animal flight (described in the "Visualization in Biomedical Research" sidebar) motivated this work. Our approach was inspired by videos of particles animating through a vessel. The video's flat view raised questions about the precise behavior of the complex 3D flow, such as how particles move in the third dimension and what their movement is relative to the vessel wall. An immersive viewing environment promised to more effectively display complex 3D structures, thus our primary challenge was to design an effective 3D visualization based on the motivational 2D visualization style.

PF tries to satisfy four goals:

- represent all flow features,
- depict flow at artery surfaces,
- allow user interaction, and
- avoid visually overwhelming the viewer.

Visualization in Biomedical Research

Our research group comprises cardiologists, bioengineers, evolutionary biologists, and computer scientists working together to test the hypothesis that the location of atherosclerotic disease is correlated with the arterial blood flow characteristics. Pathological study has shown that plaque formation is not random; thus biomedical engineers are investigating whether it's related to the local details of blood flow.[1,2] Blood flow interaction, both with substances carried in the flow and present or generated in the vessel wall, can accelerate or decelerate atherosclerosis' local progress. Understanding the interaction between the flow and the surface can help us understand the cause of detrimental medical conditions, as well as their prevention and treatment.

Computational approaches to understanding arterial flow promise to be extremely valuable tools, but are currently primitive. Real-world problem specifications and simulation models can't account for all physical components of a real flow. Furthermore, the data's scale and complexity make it difficult to understand the simulations produced.

Another interest for our group is the mechanics, dynamics, and evolution of animal flight. Our goal is to better understand how flight works in different species and how flight techniques evolved. The broader impact of our work includes the design of better flying machines. We focus on bats, which offer tremendous genetic diversity via a thousand different species. Bats are much more agile, complex, and varied than birds and fly at a greater range of speeds.

References

1. N. Woolf, *Pathology: Basic and Systemic*, W.B. Saunders, 1998.
2. W.E. Stehbens and J.T. Lie, *Vascular Pathology*, Chapman and Hall Medical, 1995.

Published by the IEEE Computer Society

This article describes the overall design of PF and discusses the system we constructed to test PF and its associated tools. We demonstrate our techniques with simulations of blood flow through an artery (an internal flow) and air flow around a bat flapping its wings (an external flow). Many other published flow visualization approaches exist (see the "Related Work in 3D Flow Visualization" sidebar) and could be complementary to PF for some data sets.

Interestingly, the choices we made in developing an effective virtual design for PF (for example, immersive viewing, the vessel wall texture, rendering thousands of haloed motion-blurred particles) are inappropriate for paper and video viewing of PF. Adapting a 2D concept to 3D or vice versa can introduce problems. To illustrate, the static monoscopic images in this article can't recreate the immersive experience of PF. A more surprising example is that if we disabled only stereo viewing in PF, viewers instantly lost their spatial sense of the flow other than that there was movement in a downstream direction.

Our approach

The goal and requirements of synoptic visualization of 3D pulsatile flow helped guide our work. Our specific approach was primarily inspired by 2D animations of particles produced by hydrogen-bubble devices and analogous physical and computer visualizations. Although these animations met a number of our requirements, they mask the flow's depth component. We thus wanted to extend this visualization style to 3D. At a high level, this involves computing particle paths and rendering them in a 3D viewing environment. This strategy requires interactive display rates, representation of all flow features, and an effective visualization design.

Precomputing particle paths as pathlines

Advecting thousands of particles is computationally intensive and impossible to accomplish interactively with current resources. Precomputing particle paths lets us animate them quickly at runtime. A path has a simulation start time and a list of points. Synchronizing advection of many particle paths at runtime is simple

Related Work in 3D Flow Visualization

Particle Flurries (PF) integrates results on visual representation from many scientific visualization sources.

Representation

Illuminated streamlines improve perception of streamlines by shading them as a function of light sources and helped motivate the particle representation and seeding in PF.[1] Fuhrmann and Groller describe a similar virtual-reality system.[2] Unlike these two approaches, we display unsteady flow, don't allow pathlines to begin or end midflow, and cycle among a much larger set of pathlines.

Analogous to Zhang et al., who carefully choose representative paths to display tensor-valued volume data,[3] we carefully choose particles for displaying 3D flow. Turk and Banks' 2D work[4] is similarly related. Like line integral convolution (LIC),[5,6] our approach aspires to show the flow's global and local behavior, but, unlike LIC, it uses time, 3D sponges, and stereo viewing.

Steinman et al.[7] produced motivational movies of carotid blood flow with 3D particle traces, but their viewpoint is outside the flow volume looking in, and their view is monoscopic. In our system, the viewer can interactively navigate inside the flow for a close-up look at flow features.

A single pathline, streamline, or streakline shows a piece of the puzzle, useful for detailed analysis. However, it's impractical for a user to interactively create and position enough streamlines and streaklines to achieve a synoptic visualization comparable to PF.

Extraction

Bryson et al.[8] point out that extracting a data set algorithmically using a precise feature description is more effective than exploring the data set interactively. To increase efficiency, PF can work with techniques that automatically extract higher-level features such as vortex cores, shocks, and recirculation.

Automatic extraction techniques don't exist for all flow features, and others are still in early development, so human-in-the-loop exploration is still needed. In some cases, the low-level particle paths can help explain particular phenomena.

Exploring flow

Several systems for exploring 3D flow exist. One of the most widely used is Tecplot (http://www.amtec.com/), a commercial desktop application. Unlike Tecplot, which runs on a conventional desktop, our approach relies on stereo, head tracking, and immersive 3D graphics to display flow data.

Bryson's virtual wind tunnel[9] is a clear antecedent to our work but requires users to manipulate widgets to generate the visualization. PF provides a complete view quickly; the user need not do more than navigate the data set.

Kuester et al.[10] implements a virtual wind tunnel in a virtual environment, rendering up to 60,000 particles at 60 frames per second. Whereas Kuester sought to enable visualization of arbitrarily large scientific data sets on commodity hardware with high graphics performance, our goal was to achieve a synoptic visualization of 3D pulsatile flow.

Max, Crawfis, and Grant developed *hairs*, a technique for visualizing flow near a surface.[11] Hairs are similar to our kelp, but we use different visual anchors and line calculations and don't allow data-driven control over colors and lengths.

Particle seeding

Like Stalling and Hege[12] and Bauer et al.[13] use quasi-random particle seeding to avoid patterns and clusters. Their approach distributes the seed points for calculating pathlines, but because it doesn't account for the pathlines

continued on p. 78

because the time difference between stored points in our implementation is constant and requires only incrementing an index into the array of path points.

Choosing a representative subset of paths

A key issue in synoptic visualization of 3D pulsatile flow is determining which particle paths to display. To represent all flow features, we need a set of pathlines that particles will follow. At a high level, we find this set by choosing a finite set of 4D seeds (that is, points in spacetime). For each seed, we compute a pathline that advects forward and backward until it exits the data set and then store it as a single pathline combining the two segments. For practical reasons, we also terminate pathlines that reach a maximum number of points. At runtime, we draw a particle first at the most upstream point of the pathline (not the seed position) and animate it until it reaches the end of the pathline.

Our goal that all features be represented requires that some particles pass through every possible feature. Because computer simulation requires a discrete description of the flow problem, a constant (possibly anisotropic) distance, D, exists below which no additional features can exist. In our arterial blood flow data set, which we computed using Nektar[1] flow codes, D is a function of the polynomial order of Nektar's spectral elements and the element density within the volume. Because particles follow pathlines, we'll meet our goal if we can compute a set of pathlines such that every point on one pathline is no further than D from a point on any other pathline.

We can meet this condition by adding seeds to fill gaps, sometimes at the cost of some redundancy in particle path coverage. We could use a simple seeding implementation that places seeds at fixed intervals of D on a regular grid, but this approach typically produces many similar pathlines. Instead, we used a Poisson-disk-based strategy for seeding. Figure 1 illustrates the technique, and Figure 2 shows pseudocode for the algorithm.

Specifically, a sweep plane steps in increments of D along the coordinate system axis that is most closely aligned with the primary flow direction. At each step, a 2D Poisson-disk seeding algorithm with radius D adds seeds to the plane approximately normal to the flow direction. (The plane already has seeds at each point where an already calculated pathline intersects the plane.) Each new plane might therefore need only a few additional seed points. We compute a pathline extending forward and backward for each new seed and then store it on disk as a list of points. Note that it's easy to use the stored data to compute the position and time of any point along the path because we record the constant integration step used and the start time of a path. This algorithm repeats until the plane reaches the end of the flow.

The implementation's data structure represents all sweep plane positions in space and time. Thus, for each

continued from p. 77

when choosing seed points, it would generate many more pathlines than our method.

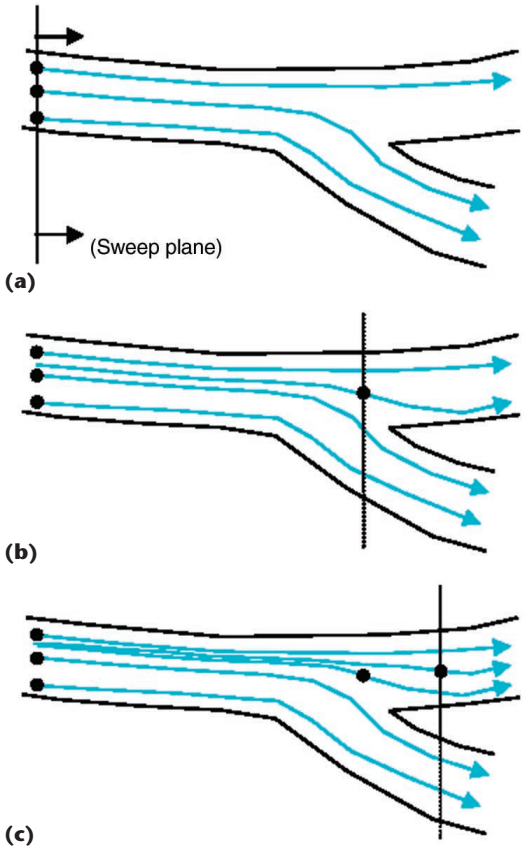
Visualization

Vis5D (http://www.ssec.wisc.edu/~billh/vis5d.html), pV3 (http://raphael.mit.edu/pv3/pv3.html), CAVE5D (http://www.ccpo.odu.edu/~cave5d/homepage.html), AVS (http://www.avs.com), and SCIRun (http://software.sci.utah.edu/scirun.html) all provide visualization functionality, but visualizations like PF haven't been produced with them, and none of the systems automatically provides a synoptic initial visualization.

Forsberg et al.[14] also construct flow visualizations, but through user-controlled widgets; our approach generates the visualization automatically.

References

1. M. Zockler, D. Stalling, and H. Hege, "Interactive Visualizations of 3D-Vector Fields Using Illuminated Streamlines," *Proc. IEEE Visualization,* IEEE CS Press, 1996, pp. 107-113.
2. A.L. Fuhrmann and E. Groller, "Real-Time Techniques for 3D Flow Visualization," *Proc. IEEE Visualization*, ACM Press, 1998, pp. 305-312.
3. S. Zhang et al., "An Immersive Virtual Environment for DT-MRI Volume Visualization Applications: A Case Study," *Proc. IEEE Visualization,* IEEE CS Press, 2001, pp. 437-440.
4. G. Turk and D. Banks, "Image-Guided Streamline Placement," *Proc. Siggraph,* Addison-Wesley, 1996, pp. 453-460.
5. B. Cabral and L. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Proc. Siggraph*, ACM Press, 1993, pp. 263-270.
6. V. Interrante and C. Grosch, "Strategies for Effectively Visualizing 3D Flow with Volume LIC," *Proc. IEEE Visualization*, ACM Press, 1997, pp. 421-424.
7. D.A. Steinman et al., "Reconstruction of Carotid Bifurcation Hemodynamics and Wall Thickness Using Computational Fluid Dynamics and MRI," *Magnetic Resonance in Medicine*, vol. 47, no. 1, Jan. 2002, pp. 149-159.
8. S. Bryson et al., "Visually Exploring Gigabyte Data Sets in Real Time," *Comm. ACM*, vol. 42, no. 8, 1999, pp. 82-90.
9. S. Bryson and C. Levitt, "The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows," *Proc. Visualization*, IEEE CS Press, 1991, pp. 17-24.
10. F. Kuester et al., "Visualization of Particle Traces in Virtual Environments," *Proc. Symp. Virtual Reality Software and Technology*, ACM Press, 2001, pp. 151-157.
11. N. Max, R. Crawfis, and C. Grant, "Visualizing 3D Velocity Fields near Contour Surfaces," *Proc. IEEE Visualization*, IEEE CS Press, 1994, pp. 248-255.
12. D. Stalling and H.C. Hege, "Fast and Resolution-Independent Line Integral Convolution," *Proc. Siggraph*, Addison-Wesley, 1995, pp. 249-256.
13. D. Bauer et al., "A Case Study in Selective Visualization of Unsteady 3D Flow," *Proc. IEEE Visualization*, IEEE CS Press, 2002, pp. 525-528.
14. A.S. Forsberg et al., "Immersive Virtual Reality for Visualizing Flow through an Artery," *Proc. Visualization,* IEEE CS Press, 2000.



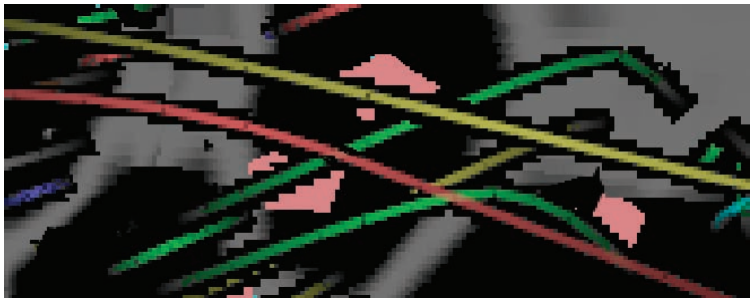
1 Simplified illustration of the Poisson-disk seeding algorithm without the time dimension. A plane sweeps from the inflow to the outflow. At each step, the algorithm adds seeds to fill gaps: (a) the three initial seeds, (b) one seed is added to fill a gap, and (c) another seed is added.

```
(1) for each slice (s)
(2)     for each timestep (t)
(3)         create seeds for (s, t) using a Poisson-disk
                distribution considering existing seeds
                marked "old"
(4)         mark each of these seeds as "new"
(5)     end

(6)     for each timestep (t)
(7)       calculate a pathline forward and backward
                for each seed in (s, t) marked as "new"

(8)       for each pathline created (p)
(9)         for each slice (s1)
(10)            determine the point of intersection (i)
                    where p intersects s1
(11)            determine the timestep (t1) at i
(12)            add i as a seed in (s1, t1)
(13)            mark this seed as "old"
(14)        end
(15)      end
(16)    end
(17) end
```

2 Pseudocode for one of our pathline calculation algorithms.



3 Close-up of the rendering of several haloed motion-blurred particles. This representation for particles reduced occlusion of distant particles, animated more smoothly, reinforced the front-to-back ordering of particles, and provided good visual cues for stereo convergence.

plane in space, multiple instances exist that differ in the instant in time the plane represents and, consequently, the set of pathlines passing through it. This is important because different pathlines can pass through a particular plane at different times, and the algorithm must know where to add seeds to ensure completeness.

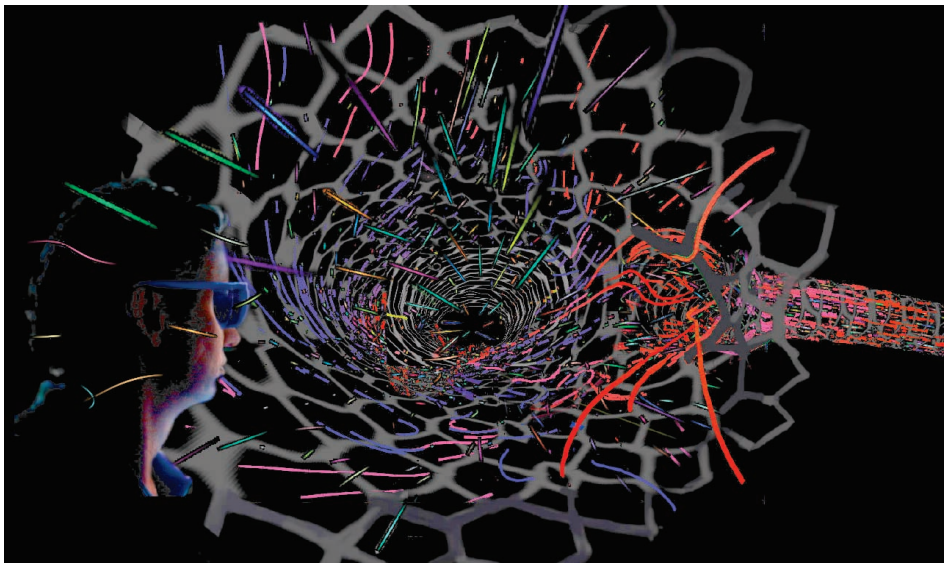
To meet a targeted distribution goal, standard Poisson-disk seeding fills a space with points by considering the placement of previous points. Our algorithm fills a 4D space with lines (or pathlines) by considering where previous lines were placed. The overhead of the pathline-set data structure and intersection and distance testing can make the Poisson-disk method slower than the fixed-interval method. However, for our artery data set and D equal to 1 unit (the artery diameter is 8 units), the fixed-interval approach produced 62,880 seeds over 16 time steps and the Poisson-disk approach produced 4,041 seeds—a 93-percent reduction.

Visual design

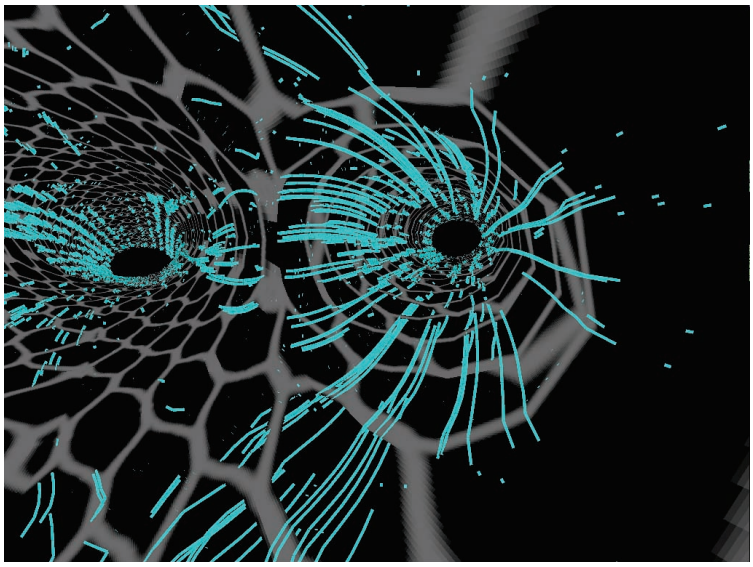
Visualizing time-varying fluid flow with animated particles requires designing a representation for particles, choosing a path for them to follow, and deciding other particle-animation details.

Representation. A particle always follows a precomputed pathline from the inflow to the outflow. We represent a particle as a motion-blurred OpenGL line-strip surrounded by a black halo, as Figure 3 shows. We create the black halo by enabling z-buffering, setting the GL line width to a thicker value (we use three pixels), turning on antialiasing, and drawing the particle lines before any other geometry. Other techniques could produce a similar effect, such as 2D texture mapping, or drawing two lines—first, a colored line with gradual alpha blending at its edges, and then a thicker black line slightly behind it.

The points defining the OpenGL line-strip are a five-vertex window on a precomputed pathline, with the vertices equally spaced in time. The center vertex has 100-percent opacity, the end points have 0-percent opacity, and the other points are interpolated between these extremes. The window advances in sync with a global timer, giving the appearance of an animated particle. We find this representation to be more effective than using a 3D geometry to represent particles because it



4 Artist-enhanced illustration of a fully immersed user in an artery just upstream of a bifurcation. The vessel wall geometry is rendered as a chicken-wire mesh to reveal its structure and the objects behind it, and to give the user a spatial reference without obscuring the pathlines.



5 Kelp in the artery shows flow information near the vessel walls. Although not animated in this illustration, the kelp near the bifurcation ebb and flow in response to the simulated heartbeat.

- reduces occlusion of distant particles,
- reinforces front-to-back particle ordering,
- provides good visual cues for stereo convergence, and
- provides a smooth motion-blurred rendering style.

We assign bright colors randomly to help differentiate individual particles.

Particle release and advection. PF maintains a set of active particles. At a user-controlled rate, particles move one step along its path. To keep the visualization synchronized, we select new particles for the set based on their initial position and start time. To address our interactive and complexity requirements, we release a user-controllable variable number of particles each frame. Fifteen additional particles per time step works well with our artery data set. Different viewers have different needs, however, so these parameters can be modified interactively. In addition, particle attributes can influence particle release behavior. For example, the viewer can release particles with equal probability or tune the visualization by increasing the probability of releasing particles with low average speed or low minimum velocity. Particle color could be mapped to other flow data quantities, but this design decision may impact the clarity of particle movement.

Tuning particle release. When we turn particle release biasing on, the release algorithm computes a normalized scalar value m ranging from 0 to 1 for each pathline. We can use several metrics—for example, we compute average velocity as $m = v/V_{max}$, where v is a particle's average velocity and V_{max} is the maximum inflow velocity. Until the algorithm has released the target number of particles for a given frame, it selects and releases a random particle if $r < (s * m)^p$ where r is a random number between 0 and 1, and s and p are interactively controlled by the user. The variable s controls the range of values between 0 and 1, and p shifts how the particles chosen are emphasized: for $p < 1$, low values are more often chosen, for $p > 1$, high values are more often chosen.

Visualizing the boundary surface. To clearly represent the curving boundary surface and let users see flow on the other side of a vessel wall, we render the geometry with a chicken-wire style texture, as Figure 4 shows. The chicken wire is composed of a thinly woven, fully opaque pattern interpolated with fully transparent sections, revealing structure without totally obscuring the flurries. Interrante et al.'s work,[2] which showed that opaque textures are more effective than semitransparent textures in conveying surface shape and interior features, inspired our choice of texture.

Flow near geometries. Flow behavior near the boundary surface geometry is interesting because it's where the flow and the surface interact. We draw *kelp* at the surface to highlight flow behavior near surfaces, as Figure 5 shows. Kelp consists of an oriented glyph at a surface anchor point and a streamline that uses a point in the flow just normal to the anchor point as its seed. A streamline is an integral path through the instantaneous flow field at a point in time, and so changes as the flow changes. We color the glyph with scalar flow data, such as pressure, and draw a streamline with length proportional to the flow's speed just normal to the anchor point

(the flow at the anchor point is always zero and therefore not useful to show). At runtime pulsatile flows cause the kelp to ebb and flow, helping to identify regions of interest—for example, locations of flow reversal or high or low shear stress. Finally, as with particle paths, we precompute the kelp seed points and streamlines to achieve runtime interactivity.

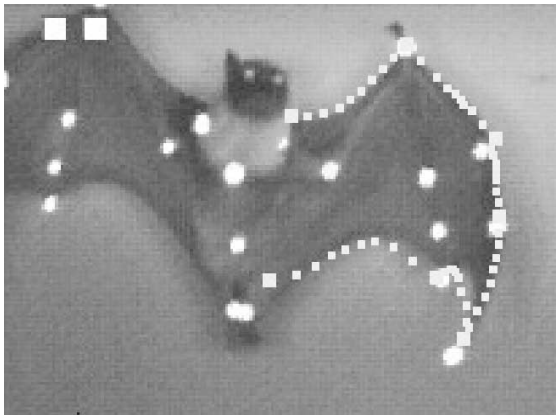
User interaction

Scientists can modify particle color and release behavior by creating virtual paint strokes, or *sponges*, directly in the data volume using the cavepainting[3] metaphor. Before creating a sponge, a user can select a color from the hue saturation value (HSV) color space using a color selector widget. Particles passing through a sponge accrue the sponge's color.

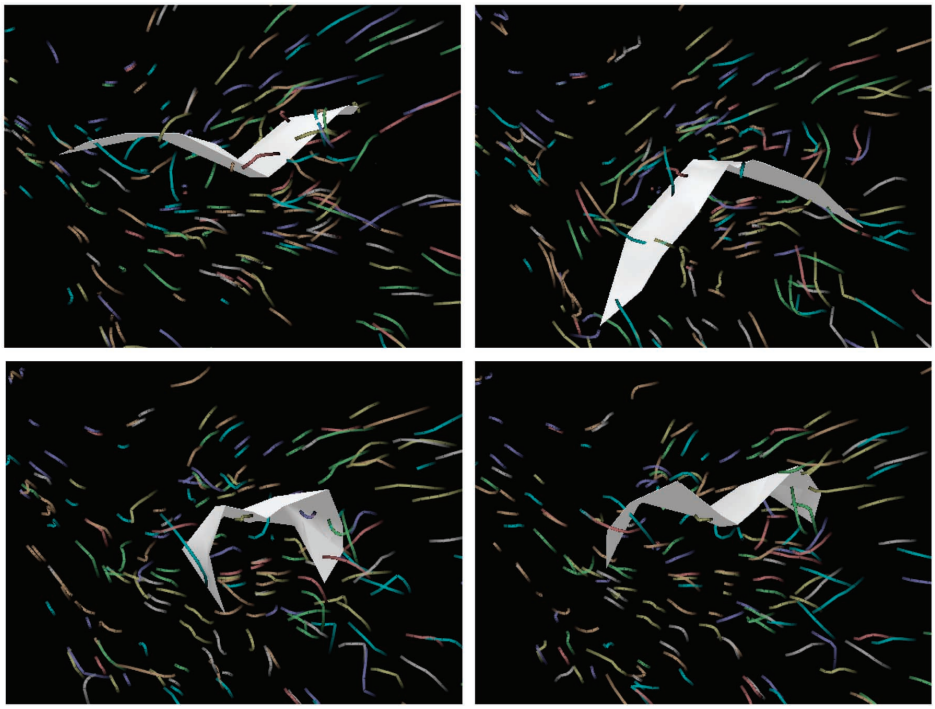
Red, white, and blue sponges have special behaviors:

- A red sponge deletes particles that pass through it.
- A white sponge acts as a particle emitter, releasing particles into the flow.
- A blue sponge prevents a particle from being released into the flow unless it will eventually pass through the sponge.

This functionality runs at interactive rates because we use a 3D lookup table the size of the data set's bounding box with a sample spacing of about 1/15th the diameter of the artery. Even this somewhat coarse sampling rate worked well. The release algorithm could quickly query the table for pointers to the set of paths that passed through the voxel containing a particular point. Deleting sponges restores deleted pathlines and resets the particles' random colors that changed via sponges. The user navigates by physically walking around the space (the head-tracked view reflects the changing viewing position), double clicking the wand button to automatically fly between stored viewpoints, using the wand to point and fly in a particular direction, or grabbing the world with a wand button press and subsequently translating and rotating the world with hand movements.



6 Snapshot of a bat flying in a wind tunnel with small reflective markers on its body. Motion capture techniques helped create a 3D bat model for use in computing data sets of simulated 3D airflow past the bat's flapping wings.



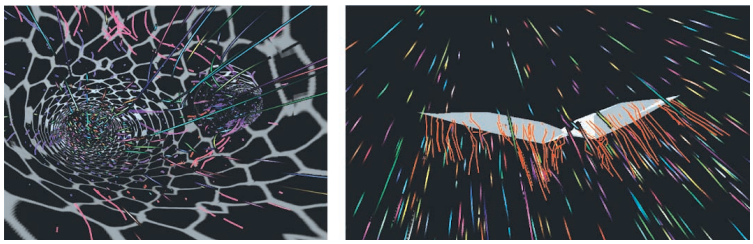
7 A time series of the bat model flying and particle flurries visualizing simulated 3D flow around it. The bat model is facing roughly toward the camera, and the primary direction of air flow is away from the camera into the distance.

Test results

We've applied our system to four flow data sets computed by Karniadakis's group's *NεκTαr* flow codes. Three were arterial blood flow data sets with varying peak inflow rates, and one was a time-varying simulation of airflow past a bat's flapping wings. We derived the arterial flow data from a prescribed idealized artery geometry and simulated it computationally.

We derived the bat flow data by motion capturing the geometry of a bat flying in a wind tunnel. We meshed a volume around the captured geometry and calculated flow velocities within that volume. Figure 6 shows an image of a flying bat with reflective markers, which helped create a 3D model of the bat. Figure 7 shows a time series of the bat model flying as particle flurries flow around it.

We implemented PF using an SGI Onyx2 driving a four-wall Cave. We achieved an average frame rate of 10 frames per second. We expect to achieve just under a 10-times speedup on commodity graphics hardware such as 3Dlabs Wildcat 6210s. Precomputation of the artery data



(a) (b)

8 **Particle flurries (a) inside an artery and (b) around a bat. Over a short time, the particle animation gives a synoptic visualization of the flow features. The kelp are the blue and pink lines attached to the artery walls. The red lines trailing off the bat help show pressure and velocity information near the geometries.**

set's pathlines and kelp required several hours and 265 Mbytes of uncompressed disk space for 50 time steps.

Domain specialists have logged tens of hours in the artery data set and are eager to spend more time exploring the flow. All viewers have appreciated the flow just by looking at and navigating through the particle flurries. Some use sponges to annotate or modify the flow to highlight specific features.

In the artery, one fluids researcher used sponges to delete flow through the center of the artery because it was less interesting; to assign different colors to side branch flow and main branch flow; to color an area of recirculation near the side branch and subsequently discover where it flowed from; to carefully study a subset of flow near the floor of the main branch; to increase the frame rate by releasing particles from midway down the main branch; to color two columns of flow near the end of the artery with contrasting colors to emphasize swirling flow; and to create a yellow sponge in the side branch to discover the shape of the area at the inflow that eventually passed through the side branch.

At certain times in the periodic flow, fluids researchers saw examples of expected flow features such as backflow in the side branch, particles reentering the main branch's flow after initially moving down the side branch, and counter-rotating vortices in the artery's curved main branch. They also found unexpected flow patterns such as a volume of space just downstream from the bifurcation in the main branch that few particles could enter at certain times of the flow, and particles that moved diagonally along the main branch wall downstream of the bifurcation when it was expected that they wouldn't have a vertical component. Finally, we found problems with the simulation parameters, such as an inadequate amount of flow moving into the side branch in an early run.

Most of our results came from the arterial blood flow data set, but we also found useful results in our air flow past a bat data set. Figure 8 shows visualizations for both data sets. From these data sets we've calculated sets of pathlines. Viewing them in the Cave has revealed some bugs in both the data and the simulations. Some were in surprising places, which PF revealed almost immediately but probably would have been hard to find using a probing or cross-section method.

Our evolutionary biologist collaborator is enthusiastic about using these tools to continue developing the motion-capture methodology, to develop and test the numerical methods for creating the data, and, ultimately, to understand and characterize wake and vortex structures, which will help us understand how different species of bats fly.

Design process

Design of PF involved a number of progressions in pathline computation, rendering, and particle distribution in the visualization. Other considerations included usability, monoscopic versus stereo viewing, and world scale. At the highest level, PF is an intuitive way to explore complex pulsatile 3D flow. It isn't, however, a stand-alone tool to reveal everything a viewer may want to see within a data set. We can combine it with a complementary visualization tool such as streaklines. Ideally, a full-featured 3D flow visualization system will include PF and many other tools.

Pathline computation

Although a fixed-interval seeding algorithm for pathline computation executes faster and is easier to implement than the Poisson-disk method, it often calculates redundant pathlines because pathlines created from seeds up- and downflow from each other can result in nearly indistinguishable paths. It can also generate many more seeds than are necessary. Our Poisson seed-spacing algorithm has reduced seed counts by 93 percent over comparable fixed-interval seeding strategies. In the algorithm's current form, the upstream slices can be oversampled because downstream seeds are integrated backward. Viewers' reports don't suggest that this is a problem, but future work might address this point.

Pathline rendering

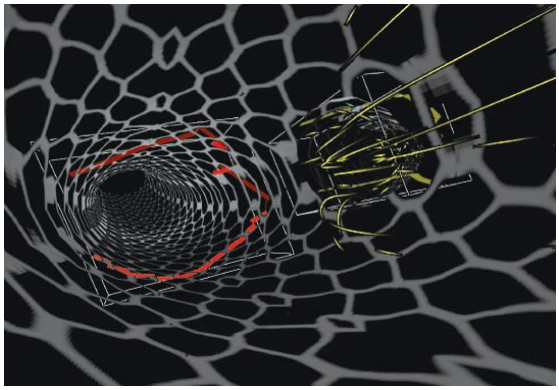
Choosing a particle animation led to surprising results. Our particle representation (see Figure 3) evolved to help clearly represent the flow in an immersive viewing environment without overwhelming the viewer. The particle used to animate a pathline didn't start as a motion-blurred, haloed, GL line, however. Initially the user could choose between a coarsely tessellated sphere, a flow-oriented triangle, or a flow-oriented textured triangle. The main problem with these geometric representations was their size, which introduced occlusion and distraction when swept near users' eyes. Also, when animated, they introduced aliasing. Although approaches to motion-blurring geometric objects exist, viewer feedback favored a line representation for particles. Although it was initially a user-controlled variable, we found most viewers favored a five-vertex window for particle representations over longer or shorter representations.

We noticed in one of the videos that inspired our work that shutter speed caused motion blur in particle movement. This led us to come up with a better line-based particle representation. Because points on a path are written out in a constant time interval, slower path segments will have many points spaced close together, and fast segments will have fewer points spread further apart. Because each particle connects the same number of points, fast particles will be longer than slow ones,

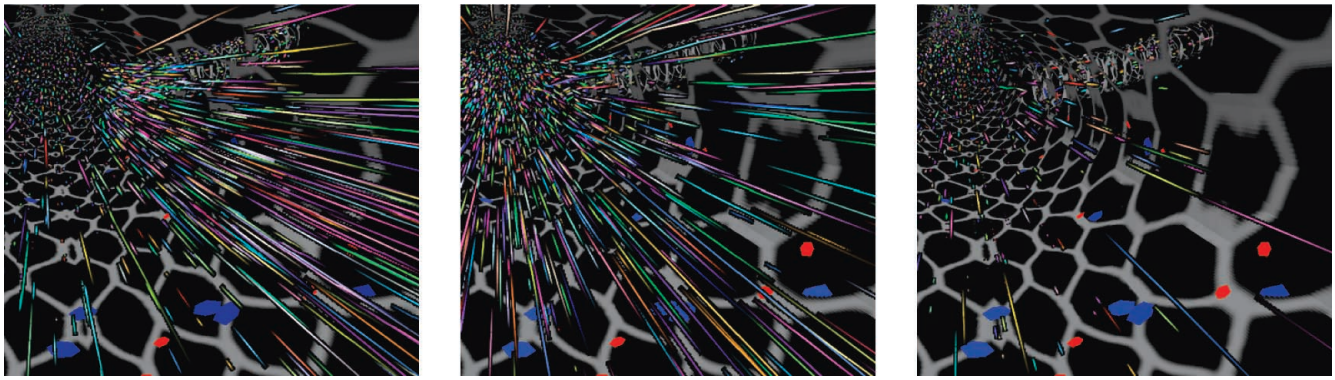
letting users make relative comparisons. Unfortunately, slow-moving particles can nearly disappear because the points become collocated. To fix this, we could always draw a particle's line representation some minimal length tangent to its pathline. We also considered inverting the effect such that slow and fast moving flow were drawn, respectively, with long and short lines. Halos enhance depth perception,[4] although we implement the halos with real-time antialiasing blending, and apply them to pulsatile flow visualization.

Particle distribution

Depending on the user's goals, particle visualization of flow data can achieve many synoptic visualizations. Initially, we released randomly colored particles into the flow with equal coverage throughout the volume. However, we extended the system to address users' desire to control particle color and release strategies as well as clearly see near-wall flow.



9 **Sponges in the artery. The red sponge on the left has deleted all flow that doesn't go through the side branch, while the yellow sponge on the right has colored all flow entering the side branch.**



(a) **(b)** **(c)**

10 **Comparison of emphasis techniques in the artery. A user can focus on pathlines that share a specific quality using emphasis. (a) The user emphasizes paths with low minimum velocity and is more likely to release particles that go through the side branch or near the walls. (b) The user emphasizes no paths and is equally likely to release all particles. (c) The user emphasizes slow paths, releasing particles near the walls more frequently.**

Sponges. Particles are immediately released throughout the vessel without any initial interaction. Users control how many particles to release at each step and the time between steps, but this only controls particles' global density. Sponges, as Figure 9 shows, let users modify particle color, release, and presence using paint strokes.

Emphasis. Emphasis controls let scientists focus on a type of pathline—for example, all slow paths—rather than all pathlines in a certain area. In practice, emphasizing slow flow has let us focus on flow running near the artery walls or entering the side branch. Emphasis also lets us focus on particles passing near the bat, which are far more interesting than the others. We can easily base emphasis on other metrics—radius of curvature, proximity to some location, or streamwise recirculation, for example. Figure 10 illustrates emphasis techniques.

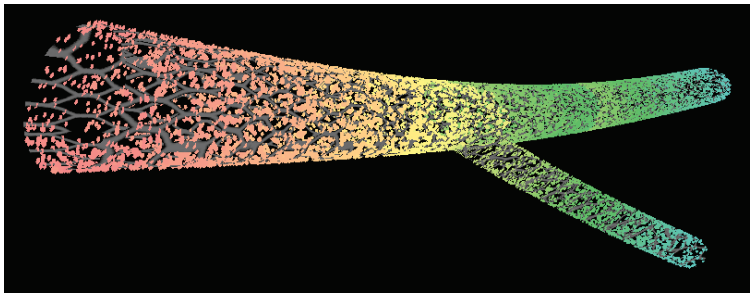
User-settable parameters and commands

Our implementation has several user-settable parameters including release rate, simulation speed scalar, emphasis type, and two emphasis-related scalar values (a scaling coefficient and a power). Three commands our implementation supports are freezing particle motion, clearing all particles, and deleting all sponges. Currently, users access these parameters and commands using a conventional keyboard on a table just inside our Cave. This approach won't scale much further, and we've considered using a command-line interpreter or a gestural or more graphical user interface technique in the future.

Particle animation only shows velocity information, ignoring other flow quantities. To show more information near the reference geometry, we initially used *splotches*, circular geometries anchored to the reference geometry and colored to reflect residence times and pressure gradients, as Figure 11 (next page) shows. We also use kelp to more directly visualize flow near the vessel wall. Some users prefer viewing the particle animation and kelp together; others prefer to view one visual element at a time but quickly toggle between the two.

Stereo viewing

Users can toggle between stereoscopic and mono-



11 **Wall splotches anchored on the artery walls. Colors along a spectrum from red to blue indicate high and low pressure. The standard technique of mapping surfaces with scalar data helps guide the viewer to potentially interesting regions to explore with PF.**

scopic viewing on the fly. In our tests, we typically started users viewing in stereo; when we toggled them to monoscopic viewing, they reported difficulty resolving the depth component of the flow and boundary geometry. When we restored stereo viewing, they could easily determine the relative depth of particles and their positions within the boundary geometry. Texturing the vessel wall with the chicken-wire-like texture also makes the wall much easier to see when viewed in stereo. The texture's hard edges provide excellent visual cues for stereo convergence whereas a smooth-shaded model appears cloudy. Making spatial judgments, such as the distance between a particle and the vessel wall, is much easier with the textured model than with a nontextured model. The texture also shows internal flow details from external viewpoints.

Dynamic world scale

We can dynamically change the scale of the world seen in PF. This can be a useful control for both internal and external flows. For external flows, it lets us scale down the data to produce an overview, or up to study a specific region. For internal flows, we scale up the data to move inside a space and comfortably view the data. For the artery data set, we generally scale the world so the artery walls appear to be 8 feet apart, letting the viewer physically move around inside it and minimizing the need for flying. Smaller diameters all but remove the need to navigate except by head movements and walking, but fusing stereo images when the user moves inside the artery vessel is difficult. Moreover, the small size makes it hard for users to appreciate the particle animation details. Larger scales, such as a 30-foot vessel diameter, decrease particles' apparent density, but also dramatically increase the amount of navigation required to see all the flow.

Conclusion

We believe PF creates an effective synoptic visualization for pulsatile fluid flow. In general, we recommend visualization designers first consider whether a synoptic visualization approach might help solve their problem better than an approach requiring more user interaction to reveal the same amount of information.

Our future work is aimed at developing a method for characterizing fluid flow visualization techniques including PF. We envision a framework in which user studies, including experts in the scientific domain and visual designers with an artistic background, would help evaluate which flow structures a particular visualization technique helps viewers see and would allow comparisons between visualization techniques. ■

Acknowledgments

This work was supported in part by the US National Science Foundation through grant CCR-0086065, and Lawrence Livermore National Laboratory Research subcontract No. B506432. This work used the Virtual Reality Peripheral Network (VRPN) library, which is supported by the US National Institutes of Health National Research Resource in Molecular Graphics and Microscopy at the University of North Carolina at Chapel Hill.

References

1. G.E. Karniadakis and S.J. Sherwin, *Spectral/hp Element Methods for CFD,* Oxford Univ. Press, 1999.
2. V. Interrante, H. Fuchs, and S.M. Pizer, "Conveying the 3D Shape of Smoothly Curving Transparent Surfaces via Texture," *IEEE Trans. Visualization and Computer Graphics*, vol. 3, no. 2, Apr.-June 1997, pp. 98-117.
3. D. Keefe et al., "Cavepainting: A Fully Immersive 3D Artistic Medium and Interactive Experience," *Proc. ACM Symp. Interactive 3D Graphics*, ACM Press, 2001, pp. 85-93.
4. V. Interrante and C. Grosch, "Strategies for Effectively Visualizing 3D Flow with Volume LIC," *Proc. IEEE Visualization*, IEEE CS Press, 1997, pp. 421-424.

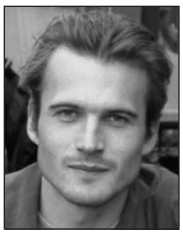
Jason S. Sobel *is a software engineer at Network Appliance in Sunnyvale, California. His research interests include scientific visualization, mobile networks, and distributed systems. Sobel has an ScM in computer science from Brown University.*

Andrew S. Forsberg *is a research staff member in the Brown University Graphics Group. His research interests include developing 2D and 3D user interfaces and virtual reality applications. Forsberg has an ScM in computer science from Brown.*

David H. Laidlaw *is an associate professor in the computer science department at Brown University. His research interests include applications of visualization, modeling, computer graphics, and computer science to other scientific disciplines. Laidlaw has a PhD in computer science from the California Institute of Technology.*

Robert C. Zeleznik *is director of User Interface Research for Brown University's Computer Graphics Group. His research interests include post-WIMP, gestural user interaction and design of 3D graphics application infrastructures. Zeleznik has an ScM in computer science from Brown.*

Daniel F. Keefe *is a PhD student in computer science at Brown University. His research interests include artistic applications of virtual reality, 3D user interfaces, and scientific visualization. Keefe has an ScM in computer science from Brown.*



Igor Pivkin *is a PhD student in the Division of Applied Mathematics at Brown University. His research interests include computational modeling of biomedical fluid dynamics, fluid-structure interaction in blood flows, and platelet aggregation. Pivkin has an MS in mathematics from Novosibirsk State University.*

George E. Karniadakis *is a professor in the Division of Applied Mathematics at Brown University and a visiting professor at the Massachusetts Institute of Technology. His research interests include fundamental numerical and theoretical research in turbulence and fluid mechanics, as well as other aspects of computational mechanics and scientific computing, in particular stochastic CFD. He has a PhD in mechanical engineering from MIT.*

Peter Richardson *is a professor of engineering and physiology at Brown University. His research interests include physiological biomechanics, including flows in blood vessels and graphic analysis from intravascular ultrasound of structure and motion in blood vessel walls. He has a PhD in engineering and a DSc in physiology from the Imperial College of Science, Technology, and Medicine of the University of London. Richardson is also a Fellow of the Royal Society of London and of the City & Guilds of London Institute, and a founding Fellow of the American Institute for Medical and Biological Engineering.*

Sharon Swartz *is an associate professor in Department of Ecology and Evolutionary Biology and an adjunct associate professor in the Division of Engineering at Brown University. Her research focuses on the mechanistic basis and evolution of animal locomotion, with emphasis on the mammalian musculoskeletal system. She has special interests in the application of physical and mathematical sciences to biological problems. She received her PhD from the Committee on Evolutionary Biology at the University of Chicago.*

Readers may contact Andrew S. Forsberg at Brown University, Dept. of Computer Science, Box 1910, Providence, RI 02912; asf@cs.brown.edu.