

# Stream Hulls: A 3D Visualization Technique for Chaotic Dynamical Systems

Kenny Gruchalla

Elizabeth Bradley

Department of Computer Science, University of Colorado at Boulder,

Boulder, Colorado 80309

gruchall@cs.colorado.edu

lizb@cs.colorado.edu

## Abstract

*This paper proposes a new three-dimensional flow volume visualization primitive called a stream hull that is specifically designed for exploring the behavior of chaotic systems. A stream hull is a surface constructed by taking the convex hull of a small “cloud” of points. A chaotic dynamical system can be visualized using stream hulls by following a large set of points as they move through the state space, breaking that set into smaller clouds of points at each time step through a nearest-neighbor search, and then constructing a convex hull of each cloud. The result is a series of volumes that illustrate the dynamics in a novel and effective manner. Stream hulls are volumetric and amorphous, naturally breaking apart in divergent areas of the system and automatically merging in convergent areas of the system, so they can provide insight into the geometry of a dynamical system and aid the viewer in understanding its flow features of the system, such as expansion, compression, twisting, and rotation. Since stream hulls are reconstructed at each time step, they adapt to the local dynamics of a system. Other visualization techniques, in contrast, either obscure the dynamics with excess information or become degenerate because of the effects of the stable and unstable manifolds upon the visualization primitives.*

## Key words

visualization algorithms, chaotic dynamics

## AMS subject classifications

76M27, 37D45

## Introduction

Many existing flow visualization techniques are poorly suited for the visualization of chaotic dynamical systems. Any graphical representation of the state of a dynamical system must handle the temporal progression of each initial condition in the system through the state space. Simply following the progression of points through time, with or without showing their paths, is a common and straightforward technique, but the long-term mixing behavior of chaotic trajectories, coupled with the projections needed to reduce their dimensionality to visualizable levels, can cause the behavior of interest to be obscured in a thicket of trajectories.<sup>1</sup> Traditional flow visualization techniques tend to enforce *a priori* correlations of points across time: that is, points used in the construction of the visualization primitive at one time step are assumed to have the same topological relationship in following time steps. Volumetric visualization techniques, for instance, rely upon deformable three-dimensional surfaces moving through the flow [1]. In nonlinear systems, where trajectories can intertwine, volume elements built from successive

---

<sup>1</sup> Indeed, the Poincaré section is an attempt to address this very issue.

points on those trajectories can invert. In a dissipative chaotic system, where the stable and unstable manifold structure elongates any initial volume along the attractor and collapses it transversely, traditional volumetric visualization techniques become degenerate; at the same time, sensitive dependence on initial conditions causes the number of visualization elements to grow exponentially. The technique presented in this paper, stream hulls, uses an adaptive volumetric primitive to solve these problems. This is useful for visualization of any three-dimensional projection of dynamical behavior.

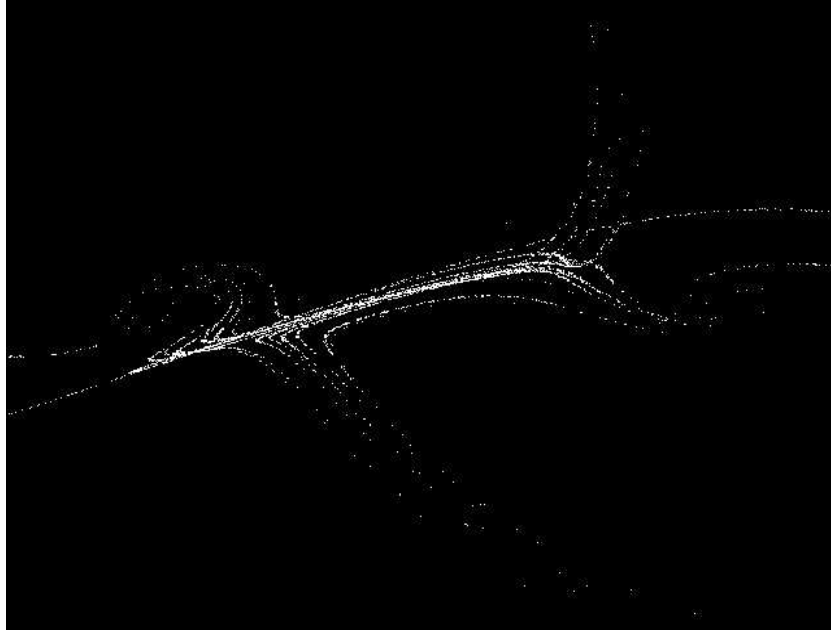
The stream hull technique is based on separating a large set of points into smaller, spatially localized clouds, and then constructing the convex hull of each cloud. As the points move through the state space of a dynamical system, the process of separating them into clouds and rendering the convex hulls is repeated. The result is a set of deformable three-dimensional surfaces that can be animated to accurately depict both local and global behavior, splitting in divergent areas of the system, merging in convergent areas, and clearly showing rotation and twisting along the way. The key here is the "renormalization" step: rather than following set initial volumes through time, with the attendant inversion/degeneracy problems described above, stream hulls are recomputed at every time step to show how the geometry of the group of tracer points deforms as they evolve through the dynamics.

### **Previous Work**

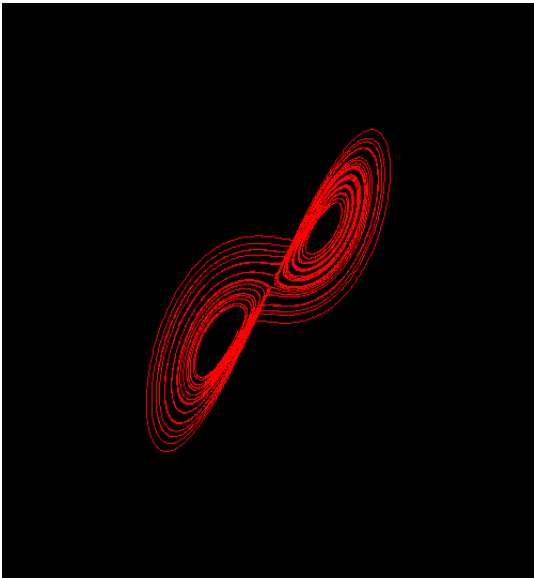
Flow visualization techniques can be separated into three classes: the visualization of characteristic elements, visualization of local properties, and direct visualization [2,3]. Characteristic-element techniques depict static representations of steady-state behavior -- for example, a periodic orbit in a dynamical system. Visualization of local properties uses glyphs to depict quantities derived from the Jacobian matrix [4]. Direct visualization, the class of approaches considered in this paper, visualizes the evolution of specific initial conditions.

One of the simplest direct visualization approaches, *surface particles*, releases a set of massless particles into a simulated flow[5]. The system's dynamics are then investigated by analyzing the instantaneous progression of the particles through the state space. This straightforward technique is directly applicable to chaotic dynamical systems, as shown in Figure 1. Surface particles are simple to implement, but provide limited insight into the expansion and compression of the state space of a chaotic system. The human eye cannot easily "store" the path of a particle, nor can it track pairs of particles easily on the many scales that are involved in such dynamics.

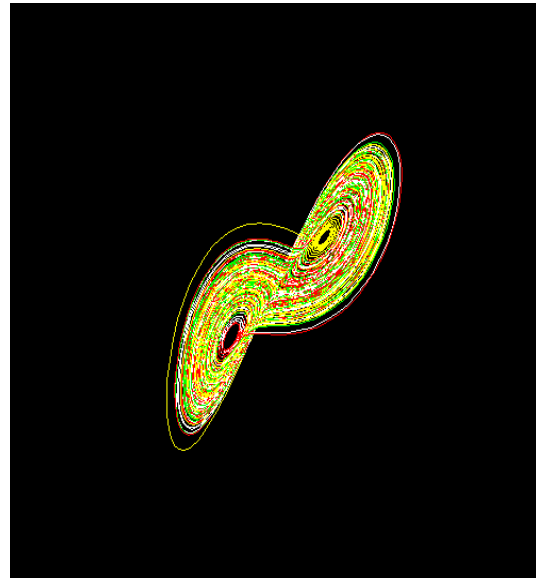
Another direct visualization primitive, the *streamline*, shows the path that a single massless particle traces through a flow field. This expands upon the idea of a surface particle by showing temporal traces, rather than instantaneous positions, of those particles. This technique can be useful for understanding dynamical systems, and is often used to construct state-space plots of attractors (see Figure 2). A single streamline, of course, shows only one trajectory through the system. Using multiple streamlines -- the classic state-space portrait -- can provide insight into the local deformation of a flow, the basin structure, etc. In a low-dimensional projection of a chaotic system, however, multiple streamlines will cross, occluding the behavior (as shown in Figure 3).



**Figure 1:** (Animation [pointsonly.mpg](#)) Surface particles shown in the Lorenz system. 5000 surface particles were densely packed on a regular three-dimensional grid. The particles were moved through the system using a fourth-order Runge-Kutta integrator.



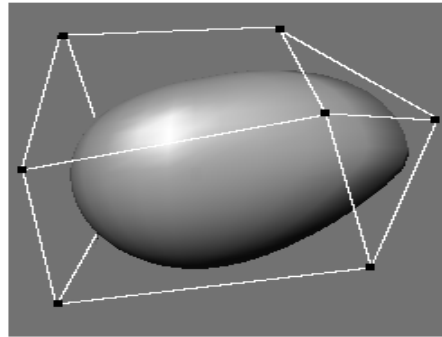
**Figure 2:** A streamline illustrating a single trajectory in the Lorenz system.



**Figure 3:** Lorenz attractor depicted with four separate streamlines. Each streamline was given a unique color (red, yellow, green, and white) and started from a unique position on the attractor. The streamlines cross in this 2D projection, occluding the behavior of the system.

Flow volumes, a higher-order visualization primitive, are typically defined as deformable three-dimensional surfaces that are swept through the system. Visualizing a chaotic system with a flow volume is of obvious interest, since higher-dimension visualization primitives can present more

properties of the system's state space than can lower-dimension visualization primitives, such as surface particles [1]. Stream bubbles [1, 6] are volumetric surfaces typically represented by NURBS (Non-Uniform Rational B-Splines) defined by a small set of control points (see Figure 4). Typically a cell from the computational grid<sup>2</sup> is used as the seed volume, and its corners are used as the NURBS control points. As these control points move through the flow, the NURBS surface volume is forced to change over time as the control points move relative to one another. In divergent flows, the stream bubble algorithm breaks the volume apart when its bounding box reaches a predefined aspect ratio. When multiple stream bubbles meet, the bubbles intersect each other; no explicit merge algorithm is used. Stream bubbles can reveal features of dynamical system's landscape such as expansion, compression, and rotation.



**Figure 4:** A stream bubble. A stream bubble is a closed NURBS surface defined by eight control points (shown as black squares).

A stream bubble's ability to split and merge as it moves through a flow suggests that this technique can smoothly handle a chaotic system's sensitive dependence on initial conditions. As control points move apart, the bubble simply splits in a natural way. Unfortunately, a direct implementation of stream bubbles is poorly suited for the visualization of a chaotic dynamical system because the closed NURBS surfaces are difficult to maintain. Points separate exponentially quickly along the unstable manifold and converge exponentially quickly along the stable one, making volume elements degenerate; when trajectories pass one another, the volume elements can even invert. For example, the lobes of the Lorenz attractor are basically flat in cross-section, so the local landscape is nearly planar. As a stream bubble moves along the Lorenz attractor, the it collapses along the transverse stable manifold, making the calculation of a closed NURBS surface numerically difficult. When a bubble moves through the "mixing region" between the two lobes, its control points can cross, which effectively inverts the closed NURBS surface. When the control points separate beyond a predefined aspect ratio, the stream bubble algorithm splits the volume element into two; because of this, sensitive dependence on initial conditions causes the number of control points to grow exponentially, and ergodic mixing distributes them across the chaotic region. Lastly, the stream bubble is a bad model of a volume in a chaotic system. The bounding volume created by the control points is misleading. This volume is described by points exterior to the closed surface, not points interior to it. In a chaotic

---

<sup>2</sup> Stream bubbles have traditionally been used to visualize computational fluid dynamics simulations which calculate flow fields over a computational grid. If the grid is 3D regular, rectilinear, or curvilinear, the grid cell is hexadral in shape with eight corner points.

system, there is no guarantee that an interior point will remain in the region bounded by the eight control points.

### **Stream Hulls**

Ideally, a volumetric visualization model for a chaotic system would be simple to render regardless of the relative orientation of the control points, the number of control points would remain constant, and the surface should be defined by points contained within the volume. The stream hull is a flow-volume technique that has these properties. It renders volumetric surfaces that evolve through the state space, much like a traditional stream bubble. Unlike stream bubbles, however, the stream hull's underlying representation “renormalizes” the volume element at every time step, showing the evolution of the local dynamics while avoiding the degeneracy and inversion issues described above. First, a large set of points are “released” in a dynamical system simulation, as in the surface particles technique. At each time step, as this set of points evolve, the set is broken into spatially contiguous groups, or “clouds,” of points. Stream hulls are then constructed by taking the convex hull<sup>3</sup> of each cloud of points, and rendering a triangulation of each convex hull with standard graphics techniques. The result is a series of small three-dimensional surfaces that move through the state space, clearly showing the geometry of the evolution of groups of nearby points.

The outline of the algorithm is as follows; details are fleshed out in the following section.

#### **Step 1. Construct an initial point set.**

Position a large number of points in the system's basin of attraction.

#### **Step 2. Group points into convex hulls.**

Select an arbitrary point from the point set. Find all the points within a predetermined radius of the selected point. Construct the convex hull of these points. Repeat this process, excluding any points already contained in a convex hull, until no unbound points exist.

The radius used to group the points defines the size of the hull. A large radius (relative to the attractor) will give a coarse global view of the system's dynamics. A small radius (relative to the size of the attractor) will provide a more accurate and detailed view into the local dynamics of the system.

The radius can also be viewed as an error bound. Stream hulls represent a volume by wrapping a set of points in surface. The stream hull surface is an approximation of how a volume would move through the state space of a dynamical system. The radius, thus, bounds extrapolation; the smaller the radius, the smaller the potential error.

#### **Step 4. Render the convex hulls.**

A convex hull is represented graphically by a closed polyhedral surface. Each polygon is shaded according to the angle between the polygon's surface normal and scene's light source, following standard graphics techniques [8].

---

<sup>3</sup> A convex hull of a set of points is the boundary, or surface, of the smallest convex domain containing the given set of points. [7]

**Step 5. Construct a new point set, by moving each point forward in time.**

The next position of each point is calculated or measured.

**Step 6. Repeat.**

Goto step 2.

For the purposes of visualizing chaotic dynamical systems, the stream hull technique has many advantages over the traditional stream bubble technique. First, the relative positions of the points do not have any ill effects on the construction of the volume; in traditional stream bubble implementation, however, the relative positions of the control points are critical to maintaining a closed surface. In the stream hull implementation, the number of points remains constant; the number of stream bubble control points moving through a chaotic system, in contrast, grows exponentially. Stream hulls are amorphous, naturally breaking apart in divergent areas of the system and automatically merging in convergent areas; the stream bubble algorithm, on the other hand, requires an explicit split operation and cannot merge bubbles.

**Results and Discussion**

As a demonstration, we applied the stream hull visualization technique to the Lorenz system:

$$\begin{aligned} \dot{x} &= a(y - x) \\ \dot{y} &= rx - y - xz \\ \dot{z} &= xy - bz \end{aligned}$$

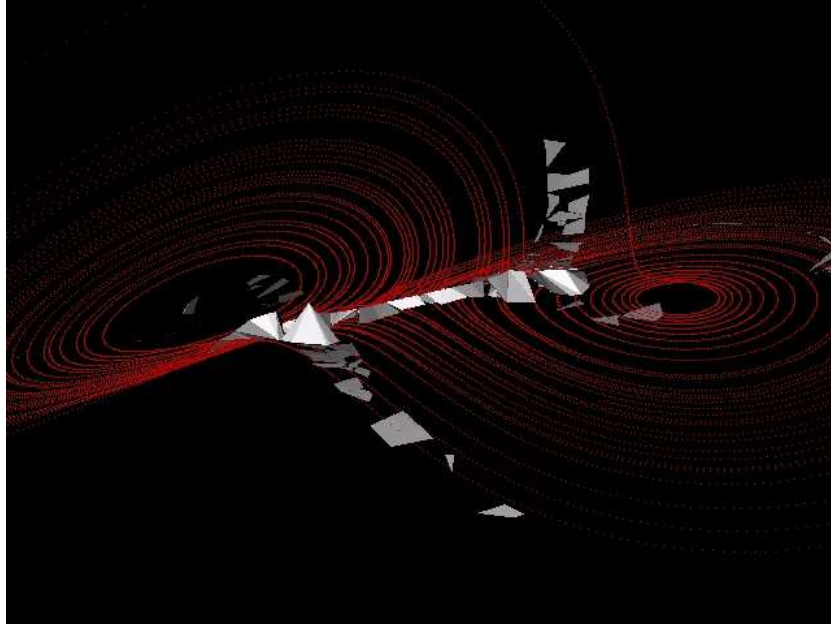
A cloud of 5000 points was integrated through the state space of the Lorenz system using a fourth-order Runge-Kutta integrator for a total of 1800 time steps.<sup>4</sup> The points were initially evenly distributed inside a small sphere located in the system's basin of attraction.<sup>5</sup> On each integration time step, a 25,000 point reference trajectory was rendered (shown in red) along with the stream hulls constructed from the 5000 points. The technique was implemented using the Visualization Toolkit (VTK), an open source object-oriented library for three-dimensional computer graphics, image processing, and scientific visualization [9]. The image of the stream hull rendering was saved at each time step to a JPEG file, these images were combined into a 1800 frame MPEG animation with a post processing step. The frames were built on 2 GHz Pentium running Linux in single user mode, at approximately 600 frames per hour. The convex hull of each cloud of points was constructed by a Delaunay triangulation. If the cloud of points was coplanar, the VTK `vtkDelaunay2D` class was used to perform the triangulation; otherwise, the `vtkDelaunay3D` class was used to construct the surface of the hull.

We generated three separate animations, varying the stream hull radius. A large radius relative to the attractor, as shown in Figure 5 and 6, provides a coarse global view of the system's dynamics. A small radius, as in Figure 7, provides a more accurate and detailed view into the local dynamics of the system.

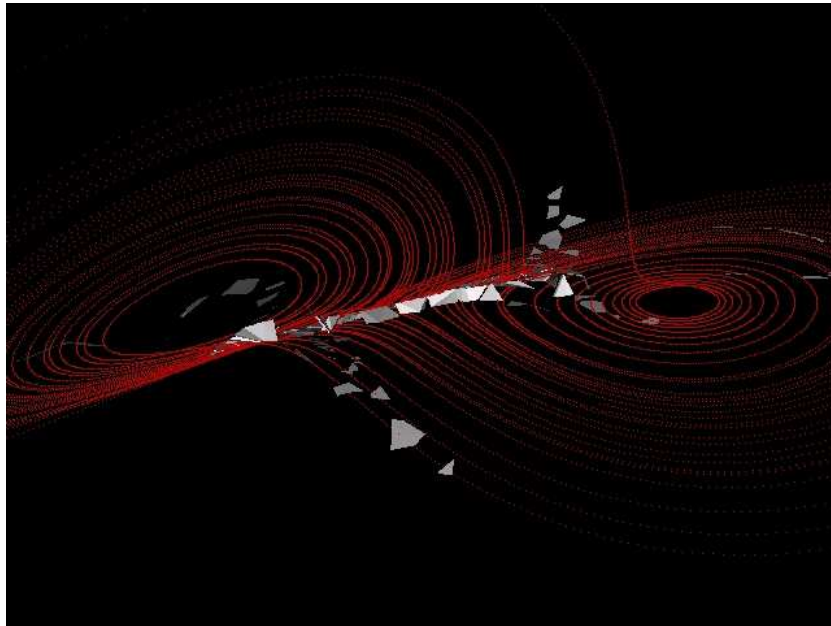
---

4 The Lorenz equation parameters used were  $a=16$ ,  $b=4$ , and  $r=45$ . The integration time step used for the Runge-Kutta integrator was 0.005.

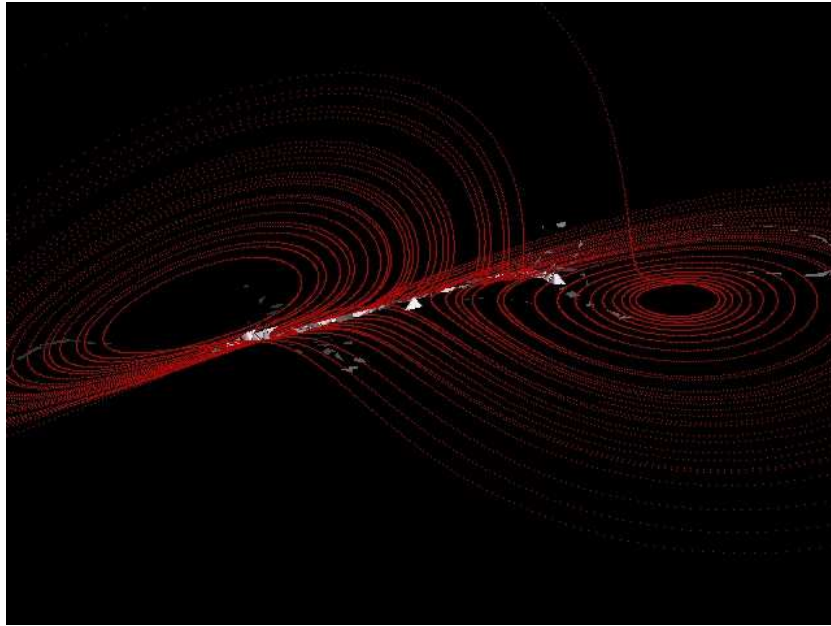
5 The initial sphere was centered on point (20, -20, 0) with a radius of 0.75.



**Figure 5:** (Animation [hull3.mpg](#)) Stream hulls with radius=3 moving through the Lorenz system. A reference trajectory is shown in red; the stream hulls are shown in shades of gray.



**Figure 6:** (Animation [hull2.mpg](#)) Stream hulls with radius=2 moving through the Lorenz system. A reference trajectory is shown in red; the stream hulls are shown in shades of gray.

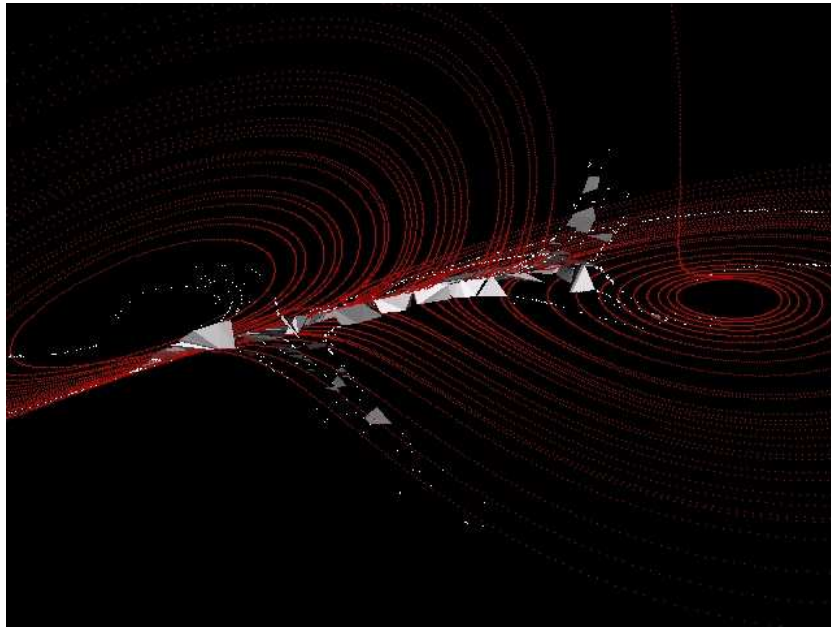


**Figure 7:** (Animation hull1.mpg) Stream hulls with radius=1 moving through the Lorenz system. A reference trajectory is shown in red; the stream hulls are shown in shades of gray.

These animations clearly illustrate the geometry of the Lorenz attractor. The planar nature of the lobes of the Lorenz attractor is apparent as the stream hulls move out from the mixing region and flatten along the attractor. Note that the shading of the convex hulls provides a useful indication of local rotation in the dynamics. The polygons comprising a stream hull's closed polyhedral surface are shaded as a function of the angle between each polygon's surface normal and the scene's light sources. Polygons whose surface normals are oriented toward a light source reflect more light and are white in appearance. As a polygon become orthogonal to the light sources, the surface becomes a darker gray conveying orientation to the viewer. This property is not unique to stream hulls; any volumetric technique has this feature. Comparing the stream hull animations (Figures 5, 6, and 7) to the surface particle animation (Figure 1), one can see the two visualization techniques produce similar results. Indeed, in the limiting case of zero radius, stream hulls and surface particles are equivalent. Both techniques illustrate the areas of convergence and divergence in the dynamics, providing insight into the mixing patterns of the state space. Stream hulls, however, also provide cues about rotation and orientation in the state space, where the surface particle technique does not. Moreover, the number of stream hulls remains roughly constant as the visualization proceeds, while the number of stream bubbles grows exponentially, which not only overloads the viewer with information, but increases the computational demands of the process.

There are conspicuous degenerate cases in our implementation of the stream hull algorithm. Not all of the 5000 points are represented by a surface in any given frame of the animations. There are two causes. First, clouds with less than three points do not contain enough information to form a surface and are discarded prior to triangulation. Second, the VTK Delaunay routines can be numerically sensitive and fail to generate a convex hull under all circumstances [9], so entire clouds of points are occasionally discarded. Figure 8 shows a frame in a stream hull animation

overlaid with the all the discarded points. Edelsbrunner has proposed the *Simulation of Simplicity* technique to cope with degenerate cases in geometric algorithms [10] which may be a way to handle the visualization of these discarded point clouds.



**Figure 8:** Stream hulls with radius=2 moving through the Lorenz system, showing the points discarded during triangulation process. A reference trajectory is shown in red; the stream hulls are shown in shades of gray, and discarded points are shown as white points.

## Conclusion

Flow visualization techniques that are designed for the visualization of computational fluid dynamics simulations do not work very well for chaotic dynamical systems. Simple techniques like surface particles or streamlines offer too little or too much information to allow the human viewer to make sense of the complex, multi-scale dynamics, while volumetric techniques can get tripped up by the stretching and folding of trajectories, and computationally overwhelmed by sensitive dependence on initial conditions.

In this paper, we introduced a new visualization technique, the stream hull, which is much better suited to the visualization of chaotic systems. It relies on a volumetric visualization element, but one that is renormalized at each time step to fit -- and accurately render -- the local dynamics.

Among the benefits of this approach are:

- The geometry of the system is clearly represented.
- The flow features such as expansion, compression, twisting and rotation are visualized effectively.
- The flow volume is based on points interior to the volume, which more accurately represents these known point values by limiting the extrapolation of the points into a volume.
- The number of control points in the visualization remain constant, rather than growing exponentially.

- The relative positioning of the control points does not complicate the construction of the flow volume.
- The merging and splitting of flow volumes is handled automatically; no explicit split and merge algorithms are necessary.

Although it is simple and well-suited to chaotic systems, the stream hull technique is computationally expensive. The less-expensive surface particle technique is nearly as effective in visualizing areas of compression and expansion in the system, but it does not show the geometry of the system, nor can it illustrate local flow features such as twisting and rotation. Streamlines convey these properties somewhat more effectively, but can overwhelm the viewer with information, occluding the behavior that is of interest. The stream-bubble technique illustrates the critical flow features without this information overload, but its volumes become degenerate in chaotic systems, and its control points multiply exponentially. The inherent renormalization of the stream-hull approach allows it to avoid all of these problems and provide an accurate and efficient picture of the local and global dynamics of a chaotic system.

## References

1. B. Zhang, and A. Pang, *Stream bubbles for steady flow visualization*, In Proceedings of the Ninth Pacific Conference on Graphics and Applications 2001, 169-177, 2001.
2. K. W. Brodile, L. A. Carpenter, et. al., *Scientific Visualization: Techniques and Applications*, Springer-Verlag, New York, 1992.
3. H. Loffelmann, and E. Gröller, *Enhancing the Visualization of Characteristic Structures in Dynamical Systems*, Technical Report TR-186-2-98-05, Institute of Computer Graphics, Vienna University of Technology, Austria, January 1998.
4. W. C. de Leeuw, and J. J. van Wijk, *A probe for local flow field visualization*, In Proceedings of IEEE Visualization '93, 1993, 117-123.
5. J. J. van Wijk, *Rendering Surface Particles*, In Proceedings of IEEE Visualization '92, 18-24, 1992.
6. B. Zhang, and A. Pang, *NURBS Blobs for Flow Visualization*, Technical Report UCSC-CRL-00-18, University of California at Santa Cruz Computer Science Department, 2000.  
[ftp:cse.ucsc.edu/pub/reinas/papers/bubble\\_nurbs.ps.gz](ftp:cse.ucsc.edu/pub/reinas/papers/bubble_nurbs.ps.gz)
7. F. P. Perparata, and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
8. J. D. Foley, A. van Dam, S. K. Feiner, and J. H. Hughes, *Computer Graphics: Principles and Practice, Second Edition in C*, Second Edition, Addison-Wesley, Reading Massachusetts, 1996.
9. D. van Heesch, (2002). VTK 4.0.2 Documentation. Retrieved October 10, 2003, from <http://public.kitware.com/VTK/doc/release/4.0/html>
10. H. Edelsbrunner and E. P. Mucke. *Simulation of Simplicity: A technique to cope with degenerate cases in geometric algorithms*, ACM Trans. Graph., 9(1):66-104, 1990.