Stream Bubbles for Steady Flow Visualization

Bing Zhang and Alex Pang
Computer Science Department
University of California, Santa Cruz, CA 95064
bing@cse.ucsc.edu, pang@cse.ucsc.edu
www.cse.ucsc.edu/research/avis/bubble.html

Abstract

This paper introduces a new relatively inexpensive technique – Stream bubbles *for 3D flow visualization. The physical analogy to this technique are bubbles that can be observed in nature with different shapes and varying speeds. A stream bubble is a surface, defined by a small set of vertices, advected through a flow field. It can easily manifest flow features like twist, stretch, expansion and rotation. Upon encountering an obstacle, stream bubbles will automatically erode and/or split in an intuitively geometric way. For highly divergent and vortical fields, it can also break apart based on the aspect ratio of the bounding volume. When two or more stream bubbles meet, no explicit merge operation is needed since the surfaces will simply intersect each other to form a composite surface. No effort is made to make the intersection smooth. Stream bubbles may be of different sizes. Larger bubbles give a coarse global view of the flow structure, while smaller ones give a more accurate depiction. In addition, our interface provides an interactive, multi-resolution visualization environment facilitated with an animated or step-by-step playback function.*

Key Words and Phrases: Subdivision, surface fairing, NURBS, flow visualization, streamline, stream ribbon, stream polygon, stream tube.

1 INTRODUCTION

In physical measurements like those from wind tunnel experiments, foreign materials (smoke, oil, or dye) are routinely introduced into the environment to observe the flow patterns [14]. However, these experiments tend to be expensive; control of environment and reproducibility of results are also issues to consider[10, 14]. On the other hand, computational fluid dynamics (CFD) simulations coupled with flow visualization techniques is gaining acceptance as a valid method for scientific investigation [11].

Flow visualization methods may either be space filling (e.g. line integral convolution (LIC) [3], spot noise [15]), or a relatively sparse sampling of the flow field. Among the techniques in the latter category are: streamlines, stream ribbons (or stream surface), stream tubes, etc. The technique presented in this paper, *stream bubble* extends the choices available in this category, but with the potential for being volume filling as well. Here are some definitions:

- A streamline is the path of a massless particle from an initial seed position within a 3D flow field. Every point along the streamline has the velocity field tangent to it.
- A stream ribbon is the path swept by a deformable line segment through a 3D flow field. For a set of discretized seeds (coming out of a rake), a stream surface can be built by tiling a series of adjacent streamlines with polygons.
- A stream tube is defined as the surface swept out by a closed polygon, or stream polygon along a streamline. The stream polygon shows an instance in "time" of the flow.
- A flow volume is defined as the space swept out by a deformable subvolume (e.g. tetrahedra). A stream bubble inside the subvolume shows an instance in "time" as the subvolume moves through the flow field.

1.1 Previous Work

We review a number of related flow visualization techniques leading up to the work presented in this paper.

There are a number of papers related to stream ribbons and stream surfaces. Hultquist [5] popularized a method where particles are repeatedly advanced a short distance through the flow field, and new polygons are appended to the downstream edge of the surface. If the ribbons grew too wide such as when in a divergent flow, the spacing of the particles were adjusted to maintain adequate sampling across the width of the ribbon. An alternate method for generating stream ribbons which does not have to worry about ribbons getting too wide is also presented in [9]. Generation of stream ribbons and stream tubes have also been improved and extended to unstructured grids [13] by carrying out the calculations in a canonical coordinate system instead of the physical coordinate system.

Instead of explicitly tracing a curve through space to generate a stream surface, van Wijk introduced two alternative ways of generating a surface-like flow structure. In [16], he defined surface-particles as very small facets, modeled as points with a normal. The shaded moving surface-particles generate a textured surface which gave an impression of the flow structure. In [17], he described a new method for constructing stream surfaces by representing it as an implicit function $f(x) = C$. By varying C, a family of stream surfaces can be generated. The idea of stream functions were also employed earlier in the efficient generation of families of streamlines [6].

Another method that also uses implicit surface representation is the stream ball technique presented by Brill et al. [2]. Each particle in the flow field is associated with a function that drops off with distance. A stream surface is defined as an isosurface over a distribution along the paths of such particles. A nice property of stream balls is their ability to automatically split or merge depending on their distance to neighboring particles.

Rather than sweeping out a point or a curve, Schroeder et al. [18] trace out an n-sided polygon through the flow field. This stream polygon is oriented normal to the local flow direction. Local deformation due to rigid body rotation and both normal and shear strain can be visualized.

Flow volumes [8] are 3D equivalent of streamlines. Tetrahedral subvolumes are swept out in space and volume rendered producing a visualization with smoke-like effects showing the path of different subvolumes through space.

These different flow visualization techniques deal with the issue of splitting (e.g. in divergent fields) and merging (convergent fields) in their own way. When two particles are headed away from each other in opposite directions, [5] assumes that there is zero-velocity critical point in between them. Thus, they will split at this critical point when the distance between them becomes too big. Likewise, they will merge when they come too close to each other. In [17], stream surfaces avoid obstacles by assigning a special small value, f_{min}, to grids within obstacles. By selecting C values above f_{min}, surfaces are guaranteed not to cross the obstacles. Stream balls handle the split/merge problem elegantly. Although, in order to get smooth stream surfaces close to the actual surface, a lot of stream balls are required followed by an isosurface extraction to form a continuous skin and is thus significantly more expensive. Curvature based adaptive subdivision of tetrahedral volumes is employed to maintain accuracy particularly in divergent flow fields [8].

1.2 Overview

Based on previous work, it is obvious that the shape (spatial distribution) and dimensionality (connectedness) of the seed points define the appearance of the resulting visualization as well as the ability to represent new features in the flow field. Individual seed points (0D) produce streamlines; seed points arranged on a rake or curve (1D) produce stream surfaces; seed points arranged to form a closed polygon (2D) produce stream tubes and polygons; seed points arranged to form a solid shape (3D) produce flow volumes or stream bubbles.

As dimension increase, stream primitives present more features than those of lower dimensions. Streamlines show fundamental properties like magnitude and flow patterns, while stream ribbons also reveal twisting motions along the local flow field. In addition to these, stream polygons (tube) show local deformations due to normal and shear strains. Because stream bubbles (flow volume) track the shape changes in a local volume of space, this technique can also present local compression or expansion of the flow field. With texture mapping, local rotation and stretching can easily be observed as well.

Since stream bubbles (flow volume) can capture more flow information, we choose to use the 8 vertices of a hexahedral cell as the initial set of seed points. A stream bubble is contained in the cell's bounding volume. We could simply observe the cell's movement in wireframe form or in polygon form, but it lacks visually pleasing effects – either lacking depth or lacking smoothness. On the other hand, air, oil, carbon-dioxide bubbles etc. are commonly observed in reality. Therefore, we use *stream bubbles* as a conservative estimate of the shape and location of the moving mass within the bounding volume of the 8 seed points. As the 8 seed points are individually traced through the vector field, the shape of the stream bubble changes with the local flow field. With only 8 control vertices, the calculation of stream bubbles is not much more expensive than stream surfaces or stream polygons. It is also less expensive than volume tracking method like stream balls [2].

2 STREAM BUBBLES

Computational fluid dynamics (CFD) simulations calculate flow fields over a computational grid or mesh. The mesh may be regular, rectilinear, curvilinear, or irregular. The first three types of mesh have cells that are hexahedral in shape. That is, each cell has 6 faces and 8 corner points. On the other hand, cells in irregular meshes are typically tetrahedral in shape with 4 faces and 4 corner points. Each of these cell types can potentially be a seed volume for starting a stream bubble. For example, if a hexahedral cell from the computational grid is selected as a seed volume, then a stream bubble is created within that volume in physical space. As the 8 corner points of the hexahedral cell are advected, the stream bubble is advanced through the volume. Similarly, a tetrahedral cell may be selected as a seed volume. In this paper, we present how a hexahedral cell is used to generate stream bubbles in steady flows.

There are several options for creating a surface, such as implicit surface, parametric surface, polygonal meshes, etc. Implicit surfaces are computationally expensive, also they are vulnerable to slight function changes and their shape is difficult to control. Parametric surfaces are widely used because of their efficient sampling and intuitive shape control mechanism. We had initially used bi-cubic NURBS (Non-Uniform Rational B-Splines) to generate a stream bubble by periodically cycling through the 8 control points. Although NURBS are very flexible, allowing for unevenly spaced knot points, closed surfaces, and additional shape control with a set of weights over each control point, they cannot deal with concave situations correctly. The left image in Figure 1 shows the corresponding NURBS bubble when the seed cell deforms to a concave volume. It is obvious that portions of the NURBS bubble is outside the bounding volume of the cell. Therefore, we turned our attention towards general purpose polygonal meshes, which can construct arbitrary topological objects. The smoothness of the surface can also be controlled by varying the number of polygonal elements.

In this section, we introduce a multi-level subdivision scheme to create a smooth, closed stream bubble surface defined over 8 control points. The generated surface is constrained meaning: (1) the surface is inside the bounding volume of the cell (note that the bounding volume is tighter than the convex hull of the 8 control points), and (2) the surface cannot penetrate itself.

The initial stream surface defined by 8 control points is constructed by first finding the center of each face of the hexahedral cell. We then form 8 triangles for each face by finding the midpoints of each edge and connecting the 4 control points and 4 edge midpoints to the face center. Hence, the initial surface is composed of 8 x 6 triangles. What we want to do next is to generate a smoother refined mesh, with the same topology that approximates this original coarse mesh.

We use a subdivision approach for mesh refinement. It basically consists of a topological split operator and a smoothing operator [7]. In order to maintain clean connectivity, we choose uniform split operations i.e. each triangle is equally divided into four higher-level triangles. Smoothing operation is surface refining or fairing. Most techniques use global energy functional to measure surface fairness, such as total curvature, thin-plate energy and membrane energy. By minimizing the global energy functional, high quality (smooth) surface can be obtained. Because discrete smoothness is intuitively characterized by low discrete curvature, we expect that the reduction in local discrete curvature by repositioning vertices will lead to global energy reduction or minimization. This task can be fulfilled by a commonly used smoothing operator, *umbrella operator*:

$$\mathcal{U}(\mathbf{p}) = \frac{1}{m}\sum_{i=0}^{m-1}\mathbf{p}_i - \mathbf{p} \tag{1}$$

Here, $\mathbf{p}_i$ is the 1-ring neighbors of vertex $\mathbf{p}$ on the mesh, m is the valence or the number of $\mathbf{p}$'s neighbors. The umbrella operator can be applied to an updating rule [12] with a damping factor $\lambda< 1$:

$$[\mathbf{p}^{n+1}] = (\mathcal{I} + \lambda\mathcal{U})[\mathbf{p}^n] \tag{2}$$

However, the basic umbrella operator might create unexpected ripples and drifting of flat surfaces. Desbrun et al. [4] provides a curvature flow scheme to smooth the surface by moving along the surface normal with a speed equal to the mean curvature. But it does not suit our constrained surface requirements because moving points along the normal might cause the repositioned vertex to go outside the bounding volume or flatten out the local subtle features. Considering that our requirement for fair smoothness is not strict, we introduce a curvature-weighted umbrella operator, and improve the updating rule by a variational damping factor to attenuate ripples and reduce unnecessary flow.

$$\mathcal{C}(\mathbf{p}) = \frac{\sum_{i=0}^{m-1} k_i\mathbf{p}_i}{\sum_{i=0}^{m-1} k_i} - \mathbf{p} \tag{3}$$

$$[\mathbf{p}^{n+1}] = (\mathcal{I} + \alpha\cdot\beta\cdot\gamma\cdot\mathcal{C})[\mathbf{p}^n] \tag{4}$$

$$\alpha = \begin{cases} +1 & \text{if } \mathbf{p} \text{ is not a concave vertex} \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

$$\beta = \mathbf{N}\cdot\frac{\mathcal{C}}{\parallel\mathcal{C}\parallel} \tag{6}$$

$$\gamma = \frac{\bar{k}+d}{1+d} \tag{7}$$

We use the divergence of the normals at vertex $\mathbf{p}$ and its neighbor $\mathbf{p}_i$ to approximate the discrete curvature k_i (normalized by π). We measure the average curvature $\bar{\text{k}}$ by summing the k_i's and dividing by m. This value is used as a scale factor for the amount of translation to be applied to the vertex. Likewise, by calculating the dot product (β) of the vertex normal $\mathbf{N}$ and $\mathcal{C}$, we can suppress some oblique drifting and totally eliminate flat surface drifting. The sign of the α factor determines the direction of translation. Whether

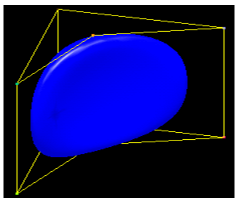

Figure 1. **NURBS bubble**

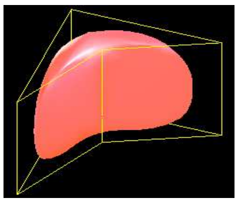

Figure 2. **Subdivision bubble**

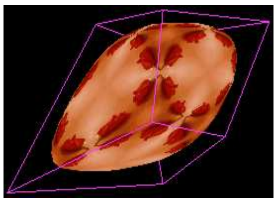

Figure 3. **Textured freeform subdivision**

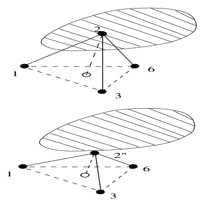

Figure 4. **Top to bottom erosion**

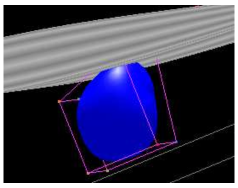

Figure 5. **Pre-erosion**

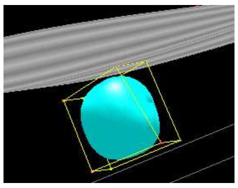

Figure 6. **Post-erosion**

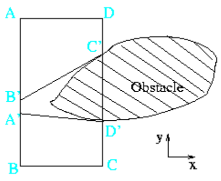

Figure 7. **2D split**

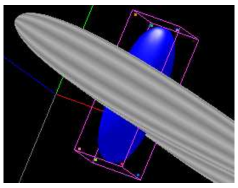

Figure 8. **Pre-split**

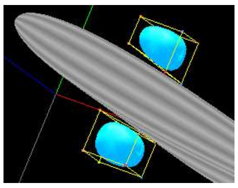

Figure 9. **Post-split**

the vertex is at a convex or concave region, the repositioning operation reduces the local discrete curvature.

Our subdivision proceeds in multiple levels to accelerate convergence. The coarser the mesh or the lower the level (larger depth d), the larger the amount of reposition during each iteration of discrete fairing. To prevent global shrinking of the shape, we also impose some boundary conditions. We distinguish among soft, hibernate, and hard vertices: Soft vertices can be updated freely; Hibernate vertices or face centers will not join the reposition until the last adaptive fairing step. They help keep the refinement stable; Hard vertices or concave skeleton vertices are always fixed to ensure that the resulting surface is always inside the bounding volume. Time complexity for subdivision is $O(2^d)$.

We currently use a pre-specified depth for subdivisions. The generated stream bubbles at level 2 are smooth enough for our purpose. The proposed scheme also properly keeps the generated surface within the bounding volume even when the bounding volume is concave. For cases when the volume is very thin, the bubble's size need to be readjusted to prevent inter-penetration. Figure 2 shows a concave subdivision bubble. Note that unlike the NURBS bubble, the subdivision bubble is entirely within its bounding volume. Figure 3 shows a textured freeform subdivision bubble. The two subdivision surfaces are at depth 2, and the total triangles rendered is 48 x 4^d for each bubble. Because the subdivision level tend to be low, the rendering cost is much cheaper than that of streamballs, which are created by implicit functions [2]. Moreover, the latter needs a large quantity of streamballs to produce a desired visualization effect.

3 FLOW IN SPECIAL REGIONS

The choice of integration for advecting the control points affects the quality of the visualization. For stream bubbles, we use an explicit fixed time step integration for two convenience reasons: (1) we are simultaneously tracking 8 control points per stream bubble and they need to be synchronized, and (2) while we are currently looking at steady state flow, using explicit fixed time step integration will facilitate extending our work to time dependent flows. Beyond these two factors, explicit integration also facilitates the handling of flow in special regions such as in obstructive, divergent or spiral fields. Here, we examine two special cases.

3.1 Flows Near Obstacles

Flows near obstacles, such as wing surfaces, are usually handled differently. For example, adaptive time step algorithms are used to prevent collision of streamlines with obstacles [1]. Because we are using fixed step integrations, we need to handle the case when some of the control vertices encounter an obstacle. In this situation, it would appear that flow must be stopped because there is no valid data to proceed with the advection. However in reality, the flow continues along the boundary of the surface. We therefore look at two geometric methods to deal with the case of missing data such as when running into obstacles. We call these the *erosion* and *split* procedures.

3.1.1 Erosion.

Erosion is a heuristic strategy based on observing flow behavior. When a bubble runs into an obstacle, part of the object is dragged, perhaps through skin friction, by the obstacle and the object is stretched or deformed. Therefore, the *erosion* procedure allows the stream bubble to slide and deform along an obstacle. Figure 4 shows one case of erosion. Additional cases of erosion models are discussed in [19].

In a hexahedral cell with 8 control vertices, each vertex has 3 immediately connected neighbors. If a vertex runs into an obstacle, it will be repositioned just outside of the obstacle by being pulled toward the center of its available outside neighbors. This process is iterative, so no vertex will collapse unless all 8 vertices are inside the obstacle, in which case, the stream bubble will be extinguished. After erosion, integration is resumed using the new set of vertex positions. Figure 5 and Figure 6 show an erosion example.

3.1.2 Split.

The *erosion* procedure allows the stream bubble to slide and deform along an obstacle, but it cannot deal with the situation where all the vertex data are valid, but the cell or volume of the stream bubble still intersects the obstacle. For such cases, we use the *split* strategy presented below.

We illustrate the splitting procedure in 2D (see Figure 7). As the cell ABCD approaches an obstacle from the left, we split the stream bubble into two parts. The natural split is along the obstacle's tangent lines of B'C' and A'D', which also better follow the obstacle's shape. The two resulting cells would then be AB'C'D and A'BCD'.

The split is similar in 3D. Instead of splitting along tangent lines, the old stream bubble would split along tangent planes of the obstacle. Split directions (x, y or z) is weighed for selection by measuring how well it would conserve the old volume after splitting. Splitting a stream bubble involves creating new vertices to go with each new cell. Because 8 old vertices are separated into two parts, 8 new vertices need to be created. For each old vertex which currently belongs to one part, a new vertex will be correspondingly implanted into the other part through a process similar to erosion. Details of this process are discussed in [19]. Figure 8 and 9 show a split example.

3.2 Flows in Divergent or Vortical Regions

Let us have a look at situations where the stream bubble is stretched excessively e.g. in high shear regions, or curved excessively e.g. near vortices. When massless particles enter a divergent or repelling area,, or when they enter a spiraling course, nearby particles may end up quite some distance relative to each other. In such situations, a single stream bubble would not be able to represent the local flow features. Therefore, we apply a *break* action that breaks up a stream bubble into two pieces.

For both situations, the *break* procedure is straight forward. We simply cut the bubble in half. We check if both the bubble's size and the ratio of the bubble's longest axis and shortest axis is over some interactively set threshold. If so, then a break will happen along the center plane whose normal is parallel to the longest axis. The setting of additional vertices is similar to the *split* procedure.

4 USAGE AND RESULTS

Massless streamlines can not tell much about local twisting and stretching without depth information; similarly, flat stream ribbons can not tell much about local volume expansion and rotation. More streamlines and stream ribbons can only clutter images. Stream tube manifests flow structure, but it can not tell flow detail. Figure 10 shows streamlines from four neighboring vertices, two adjoining stream ribbons and a stream tube of a stream bubble. On the other side, flow features can be easily observed through expansion/compression, twisting/rotation, and erosion/split/break of color or texture mapped stream bubbles. Because stream bubbles generalize earlier work, our system can easily add more visual impressions by superimposing other visualization primitives such as streamline representation from individual vertices, stream ribbon representation from pairs of vertices, and stream polygon representation from faces of the hexahedral cell. In addition, the system supports static views, step-by-step animations, and continuous playback of the evolution of stream bubbles. Figure 11 shows a sequence of the framed and texture mapped stream bubble flowing in the superimposed stream tube. Combined with static flow features, it clearly demonstrates the deformation, contraction and rotation of the stream bubble.

When investigating a flow field, we put one or more stream bubbles in it. The size of the stream bubbles may be different. Smaller ones allow more detailed feature examination, while larger ones present a gross depiction of the steady flow field. The size, shape, and position of volume seeds can be adjusted interactively. If the flow field is highly turbulent such that the stream surface attempts to intersect itself, the system will issue a warning and the cell needs to be broken down into smaller pieces. Because the 8

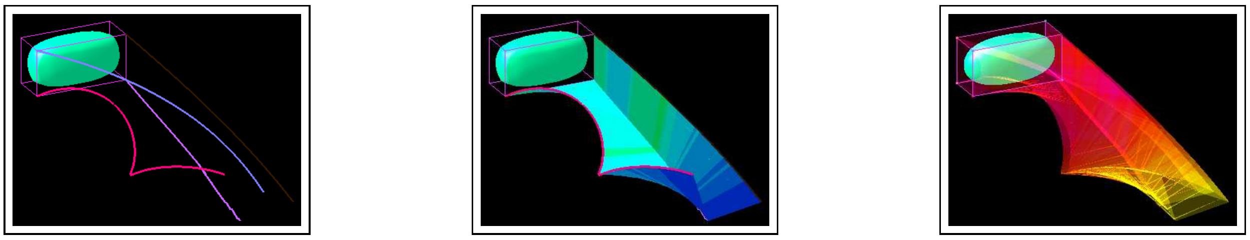

Figure 10. Left: streamlines from four neighboring vertices, each is uniquely colored; Center: two adjoining stream ribbons, color-mapped to relative volume; Right: the stream tube, reverse color-mapped to stream volume.

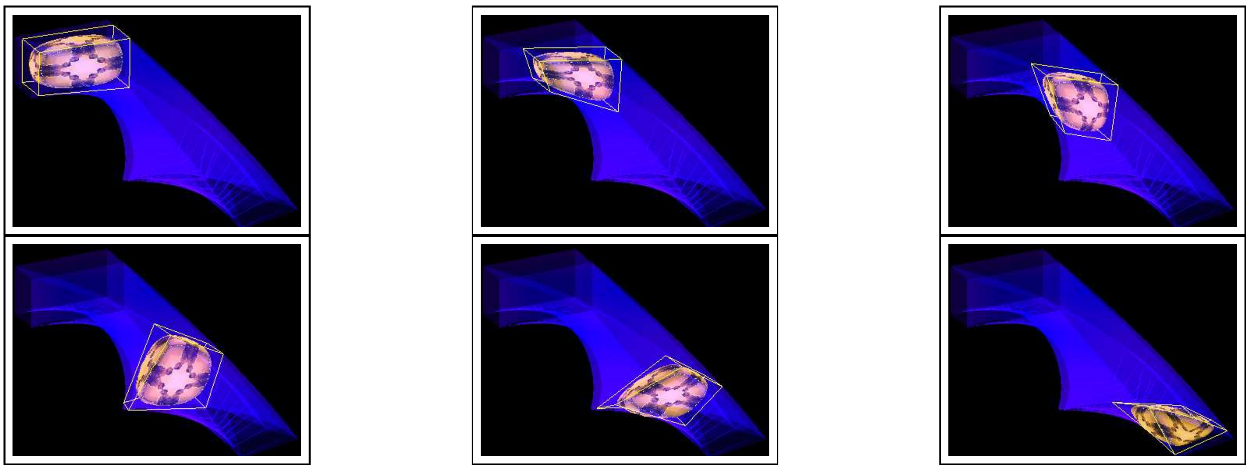

Figure 11. Framed and textured stream bubble flowing in the superimposed stream tube (up-down, left-right). It clearly demonstrates flow features like deformation, contraction and rotation.

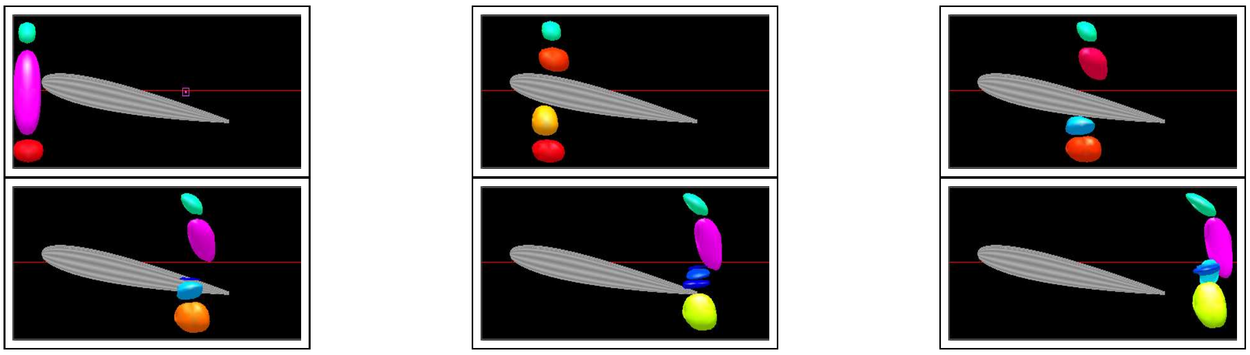

Figure 12. Three different stream bubbles flowing around the wing from the same start line on the left. The big center stream bubble undergoes a split process. The bottom half undergoes further splitting and erosion and produced stream bubbles that twist and rotate more radically. Other stream bubbles are comparatively steady. Color is mapped to the velocity magnitudes of stream bubbles.

control points of each stream bubble are traced individually, two stream bubbles may collide and penetrate through each other. If they do, no special handling is necessary as the two stream bubbles will simply appear to merge. Of course, since they are really computed separately, they can just as easily go their own ways again. Figure 12 shows a time

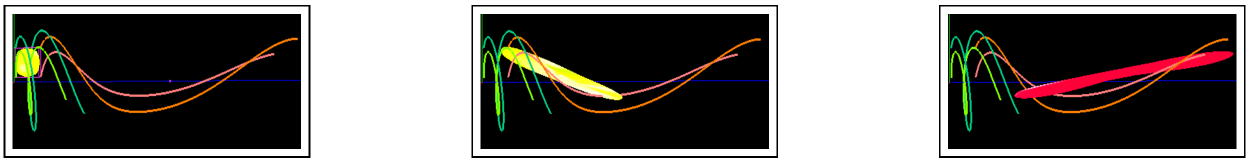

Figure 13. Flow in vortical field without *break* process. Color is mapped to the volume of stream bubble. As the stream bubble rotates in the field, it also stretches along the Z axis (increasing to the right), and finally occupies almost the entire field. The stream bubble incorrectly cuts across the field and misses regions of high curvature.

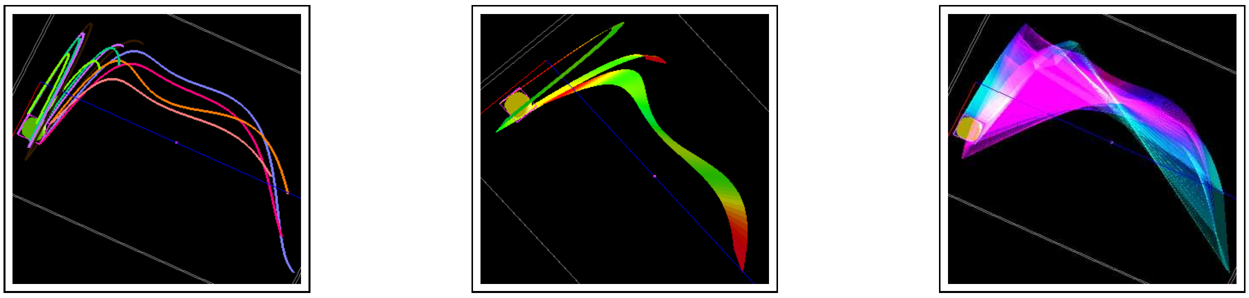

Figure 14. Stream primitives for Figure 13's flow from a different viewpoint. Left: streamlines from 8 vertices of the stream bubble; Center: two neighboring stream ribbons; Right: the stream tube. The correct flow structure is not evident in these variations. In contrast, using stream bubbles, Figure 15 shows the flow structure clearly.

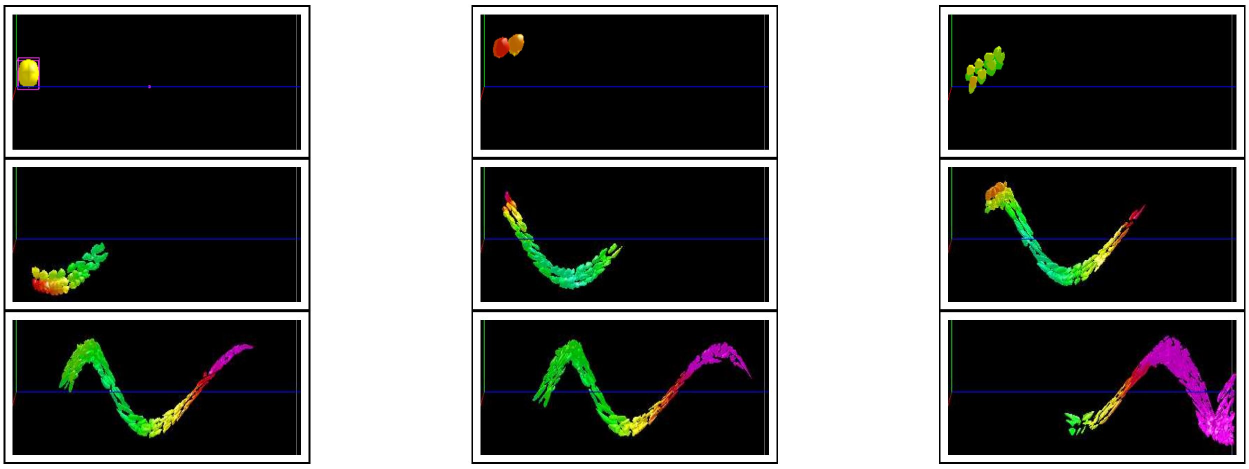

Figure 15. Flow in the vortical field with *break* process. Color is mapped to the velocity magnitudes of stream bubbles. Stream bubbles keep breaking while swirling around and moving along the Z-axis (to the right) with different speed. The color of the stream bubbles show that speed increases to the right and away from the axis of rotation. The curvature of the flow is clearly manifested in the chain of stream bubbles.

series of three stream bubble starting with different sizes from the same line, moving around a stationary, gray wing. In the beginning, the big center stream bubble undergoes a split process. Then the top half mainly rotates and deforms, and the bottom half slightly erodes while it flows under the wing. Later, the bottom half undergoes several splits which produce stream bubbles that twist and rotate more radically. The other two stream bubbles are comparatively steady with only some rotation and deformation. From their relative positions, it looks like there is not much speed difference, but the stream bubbles close to the wing are in fact faster because they are slowed down a little bit by erosion. The color of the stream bubbles are mapped to the the volume of the stream bubbles. Other color mappings (relative volume change, velocity magnitude, etc.) and texture mapping are also available as options.

Figure 13 shows a vortical flow field with no obstacles. As the regular stream bubble rotates in the field, it also stretches along the Z axis (increasing to the right), and finally occupies almost the entire field. Without a break operation, a single stream bubble cannot show the local flow structure correctly. In particular, the stream bubble cuts across and misses regions of high curvature. Because we are using only 8 control points per stream bubble, the stream bubble must be broken down into smaller pieces so that the overall curvature can be better represented. Figure 14 shows the corresponding stream primitives from a different viewpoint when the stream bubble does not break. Streamlines are seeded from the 8 vertices of the stream bubble. From the image, we see that four streamlines are swirling more while others are stretching more. But we can not tell the flow structure. Similarly in the center image, one stream ribbon is curling around while the other neighboring stream ribbon is extending. Additional streamlines or stream ribbons will not help. They are just as obscure as the stream tube on the right. Figure 15 displays the breaking sequence. Color is mapped to the velocity magnitude of the stream bubble. In the beginning, the stream bubble breaks into two stream bubbles. Then they keep breaking while swirling around and moving along the Z-axis (to the right) with different speeds. Stream bubbles on the right and farther away from the Z-axis are moving faster than those on the left and closer to the Z-axis. From the image sequence, we can understand the flow structure much better. The curvature of the flow is manifested in the chain of stream bubbles.

5 CONCLUSIONS

In summary, we introduced the concept of *stream bubble* for flow visualization. Among its benefits are:

- ability to show flow features such as expansion and compression, twisting and rotation;
- compact representation for a large portion of the flow volume which allows for erosion along an obstacle surface and splitting against an obstacle; or break in highly divergent field or spiral field;
- since each stream bubble is tracked separately, there is no need to explicitly merge upon contact with another stream bubble;
- it generalizes a set of other flow visualization techniques – streamline representation from individual vertex, stream ribbon representation from pairs of vertices, and stream polygon representation from faces of the hexahedral cell.

There are a number of things we are currently working on to improve stream bubbles. The list includes: investigation of the use of weights to facilitate the splitting process; support adaptive subdivisions at high curvature regions in order to minimize space and time requirements; extensions to non-hexahedral seeds so as to support flow data defined over unstructured meshes; and extension to time dependent flow visualization.

ACKNOWLEDGMENTS

We would like to thank David Kao for discussions on flow fields, Patrick Moran and Chris Henze for their help with FEL, and Jennifer Dacles-Mariani and Greg Zilliac for their data sets. We would also like to thank members of the Advanced Visualization and Interactive Systems laboratory at UC, Santa Cruz for their suggestion and help. This project is supported by NASA grant NCC2-5281, LLNL Agreement No. B347879 under DOE Contract No. W-7405-ENG-48, DARPA grant N66001-97-8900, NSF NPACI ACI-9619020, and ONR grant N00014-96-1-0949.

References

[1] FAST home page. science.nas.nasa.gov/Software/FAST.

[2] Manfred Brill, Hans Hagen, Hans-Christian Rodrian, Wladimir Djatschin, and Stanislav V. Klimenko. Streamball techniques for flow visualization. In *Proceedings of Visualization 94*, pages 225–231. IEEE Computer Society, 1994.

[3] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings SIGGRAPH*, pages 263–270, Anaheim, CA, August 1993. ACM SIGGRAPH.

[4] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings SIGGRAPH 99*, pages 317–324. Addison Wesley, 1999.

[5] J.P.M. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proceedings: Visualization '92*, pages 171–178. IEEE Computer Society, 1992.

[6] D. N. Kenwright and G. D. Mallinson. A 3-d streamline tracking algorithm using dual stream functions. In *Proceedings: Visualization '92*, pages 62–68. IEEE Computer Society, 1992.

[7] Leif P. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer*, 16(3):142–158, 2000.

[8] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proceedings of Visualization 93*, pages 19–24. IEEE, 1993.

[9] Hans-Georg Pagendarm and Frits H. Post. Studies in comparative visualization of flow features. In G. Nielson, H. Hagen, and H. Muller, editors, *Scientific Visualization: Overviews, Methodologies, Techniques*, pages 211–227. IEEE Computer Society, 1997.

[10] Qin Shen, Alex Pang, and Sam Uselton. Data level comparison of wind tunnel and computational fluid dynamics data. In *Proceedings of Visualization 98*, pages 415–418, 557, 1998.

[11] Paul Smith and John van Rosendale. Data and visualization corridors: Report on the 1998 dvc workshop series. Technical Report CACR-164, California Institute of Technology, September 1998.

[12] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings SIGGRAPH 95*, pages 351–358. Addison Wesley, 1995.

[13] S.K. Ueng, C.Sikorski, and K.L. Ma. Efficient streamline, streamribbon, and streamtube constructions on unstructured grids. *IEEE Trans. Visualization and Computer Graphics*, 1(3):210–217, 1996.

[14] Samuel P. Uselton. exVis: Developing a wind tunnel data visualization tool. In *Proceedings of Visualization 97*, pages 417–420. IEEE, 1997.

[15] J. J. van Wijk. Spot Noise: Texture synthesis for data visualization. *Computer Graphics*, 25(4):309–318, 1991.

[16] J. J. van Wijk. Rendering surface particles. In *Proceedings: Visualization '92*, pages 54–61. IEEE Computer Society, 1992.

[17] J.J.Van Wijk. Implicit stream surfaces. In *Proceedings: Visualization '93*, pages 245–252. IEEE Computer Society, 1993.

[18] W.Schroeder, C.Volpe, and W.Lorensen. The stream polygon: A technique for 3D vector field visualization. In *Proceedings: Visualization '91*, pages 126–132. IEEE Computer Society, 1991.

[19] Bing Zhang and Alex Pang. Nurbs blobs for flow visualization. Technical Report UCSC-CRL-00-18, UCSC Computer Science Department, 2000. ftp.cse.ucsc.edu/pub/reinas/papers/bubble_nurbs.ps.gz.