NURBS Blobs for Flow Visualization

Bing Zhang and Alex Pang

UCSC-CRL-00-18
November 27, 2000

Jack Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

This paper introduces a new relatively inexpensive technique – *Stream bubbles* for 3D flow visualization. The physical analogy to this technique are bubbles (air, carbon-dioxide, oil, etc.) that can be observed in nature with interesting shapes and varying speeds. A stream bubble is represented by NURBS with only 8 control vertices, although other representations are also possible. Its internal shape changes over time and shows twisting and stretching, as well as expansion and contraction along the flow. Upon encountering an obstacle, it will automatically erode and/or split in an intuitively simple geometric way. For highly divergent and vortical fields, stream bubbles 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. Bubbles may be of different sizes. Large bubbles give a coarse global view of the flow structure, while smaller bubbles give a more accurate depiction of the local flow structure. In addition, our interface provides a good view of the flow structure by allowing users to interactively move and resize different stream bubbles together with an animated or step-by-step playback facility.

Keywords: Vector field, NURBS, streamline, advection, splitting, expansion, contraction.

1 INTRODUCTION

In physical measurements such as those from wind tunnel experiments, foreign materials such as smoke, oil, or dye are routinely introduced into the environment to better observe the flow patterns [13]. However, these experiments tend to be expensive; control of environment and reproducibility of results are also issues to consider [10, 13]. 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) [4], spot noise [14]), 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, extends the choice 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 along a streamline. A stream polygon shows an instance in "time" as a closed polygon is being swept out along a streamline.
- A flow volume is defined as the space swept out by a subvolume (e.g. tetrahedra). A stream bubble shows an instance in "time" as a subvolume is being swept out. It is defined as a closed surface along the path swept out by a set of seed points initially defining a subvolume of the 3D flow, and shows the evolution of that subvolume as it goes through the 3D flow.

While streamlines may reveal the magnitude and flow patterns, stream ribbons show twisting motion along the local flow field. In addition to these, stream tubes also show local deformations due to normal and shear strains. Because stream bubbles track the shape changes in a local volume of space, this technique can also show local compression or expansion of the flow field. With texture mapping, local rotation and stretching can be easily observed as well. Computationally, it is also less expensive than volume tracking method like stream balls [2]. These different techniques are reviewed in the next section.

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 [8]. Generation of stream ribbons and stream tubes have also been improved and extended to unstructured grids [12] 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 [15], 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 or path 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 [7] 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 field) and merging (convergent field) 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 low f 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 [7].

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. 0D or individual seed points produce streamlines. 1D or seed points arranged on a rake or curve produce stream surfaces. 2D or seed points arranged to form a closed polygon produce stream tubes and polygons. 3D or seed points arranged to form a solid shape produce flow volumes or stream bubbles.

Just as stream surfaces represent additional flow features (e.g. twisting) than streamlines, and just as stream polygons can show yet additional features (e.g. normal and shear strain) than stream surfaces, stream bubbles can show new features too (e.g. expansion and contraction of the local flow field, and local rotation).

Our current implementation of stream bubbles uses the 8 vertices of a hexahedral cell as the initial set of seed points. The bubble itself is contained within this bounding cell volume and is represented by a NURBS surface. We chose this type of surface because of its convex hull property. The stream bubble is a conservative estimate of the shape and location of the moving mass within the convex hull 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. Stream bubbles may be small or large. Smaller bubbles allow more detailed examination of the flow, while larger bubbles represent a gross depiction of the flow field.

Because the 8 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 appear to merge. Of course, since they are really computed separately, they can just as easily split into two again.

Section 2 explains how we construct a stream bubble with 8 control points using a NURBS surface. This is followed by a discussion in Section 3 of how stream bubbles are deformed and split when they encounter obstacles, or broken when the flow is highly divergent and stretches or bends the bubbles too much. In Section 4, we describe our implementation and some results. We summarize our findings and give a list of improvements that we are working on in Section 5.

2 NURBS AND STREAM BUBBLES

Computational fluid dynamics (CFD) simulations calculation 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.

There are several options for creating a surface defined by 8 control points. Specifically, one can select from an array of surface definitions such as Catmull-Rom, Bezier, NURBS, etc., as well as the degree or smoothness of the surface. Since the stream bubble is to represent the mass bounded by the 8 control points, an important consideration is the convex hull property of the surface. For this paper, we consider stream bubbles defined as bi-cubic NURBS surfaces.

2.1 NURBS

NURBS or Non-Uniform Rational B-Splines surfaces are quite flexible, allowing for unevenly spaced knot points, closed surfaces, and additional shape control with a set of weights over a small number of control points. The NURBS surface $Q(u, v)$ defined over the parameters u and v with m and n control points along each respective parameter, given the control points $P_{i,j}$ and corresponding weight $w_{i,j}$ is defined in Equation 2.1 below. The product of the B-Spline basis functions $M_{i,d}$ and $N_{j,d}$, both of order d, form the tensor basis function used by the NURBS surface. The Cox-deBoor formulation for the B-Splines functions are presented below as well [9, 16].

$$Q(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} P_{i,j} M_{i,d}(u) N_{j,d}(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} M_{i,d}(u) N_{j,d}(v)} \tag{2.1}$$

$$\begin{aligned} M_{i,1}(u) &= \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ M_{i,d}(u) &= \tfrac{(u-u_i)M_{i,d-1}(u)}{(u_{i+d-1}-u_i)} + \tfrac{(u_{i+d}-u)M_{i+1,d-1}(u)}{(u_{i+d}-u_{i+1})} \end{aligned} \tag{2.2}$$

$$\begin{aligned} N_{j,1}(v) &= \begin{cases} 1 & \text{if } v_j \le v < v_{j+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{j,d}(v) &= \tfrac{(v-v_j)N_{j,d-1}(v)}{(v_{j+d-1}-v_j)} + \tfrac{(v_{j+d}-v)N_{j+1,d-1}(v)}{(v_{j+d}-v_{j+1})} \end{aligned} \tag{2.3}$$

2.2 Stream Bubble

The bounding *volume* or *cell* of a stream bubble is defined by 8 vertices [0..7], as illustrated in Figure 2.1. The axes u, v, w are defined at the geometric center of the cell. Note that over the course of a stream bubble's evolution, the cell will deform and the axes may not be orthogonal to each other. The 6x6 matrix in Figure 2.2 shows the winding sequence of control points needed to get a smooth closed surface. Figure 2.3 shows a stream bubble and its bounding volume, while Figure 2.4 shows the wireframe rendering. Figure 2.5 shows two inter-penetrating stream bubbles.

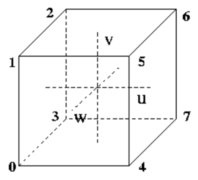


Figure 2.1: Stream bubble cell

$$\begin{bmatrix} 0 & 1 & 6 & 7 & 0 & 1 \\ 4 & 5 & 2 & 3 & 4 & 5 \\ 7 & 6 & 1 & 0 & 7 & 6 \\ 3 & 2 & 5 & 4 & 3 & 2 \\ 0 & 1 & 6 & 7 & 0 & 1 \\ 4 & 5 & 2 & 3 & 4 & 5 \end{bmatrix}$$

Figure 2.2: Winding sequence

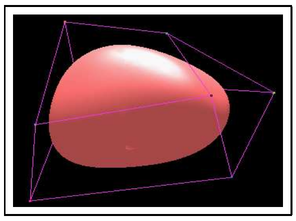


Figure 2.3: Stream bubble

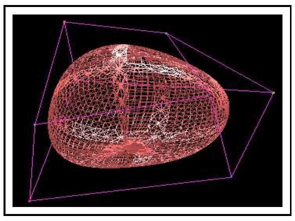


Figure 2.4: Wireframe

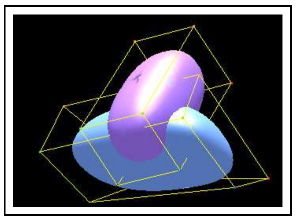


Figure 2.5: Two stream bubbles

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 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 divergent or vortex fields. Here, we examine two special cases.

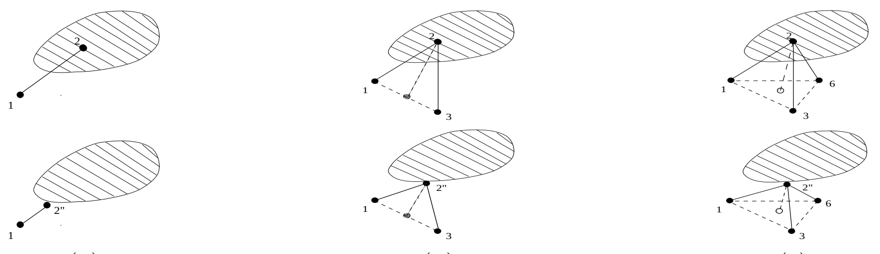
3.1 Flows Near Obstacles

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 or into a boundary region. We call these the *erosion* and *split* procedures.

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.

Figure 3.1 shows three cases of erosion. Since the stream bubbles in hexahedral cells have 8 control vertices, each vertex has 3 immediately connected neighbors. For example, the neighbors of vertex 2 in Figure 2.1 are vertices 1, 3, and 6; each pair is connected by a solid line. In case (a), vertex 2 is inside an obstacle and only 1 of its neighbors is outside the obstacle. When this case is encountered, vertex 2 is moved along the edge with vertex 1 until it is just outside the obstacle to position 2". In case (b), two of vertex 2's neighbors are outside the obstacle. For this case, vertex 2 is moved to 2" just outside the obstacle in the direction that is half way between vertex 1 and 3. The corresponding edges are also adjusted accordingly. In case (c), vertex 2 is moved towards the center of vertex 1, 3, and 6. Integration is resumed using the new set of vertex positions. Figures 3.2 a1 and a2 show an example of how a stream bubble is eroded.

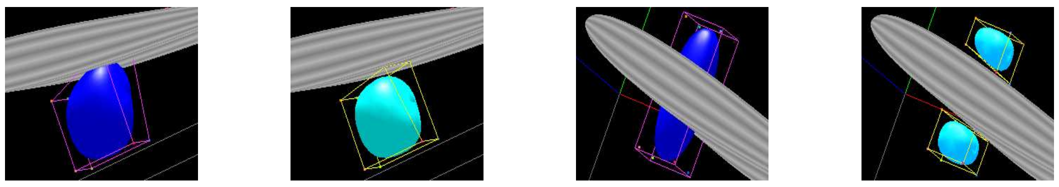


(a) (b) (c)

Figure 3.1: Erosion procedure

Split

While the *erosion* procedure allows the stream bubble to slide and deform along an obstacle, it cannot deal with the situation of invalid center data of neighbors (e.g. in Figures 3.1 b and c where the midpoint or center of the neighbors is still within the obstacle), or the situation where all the vertex data are valid, but the cell or volume of the stream bubble still intersects the obstacle. For these cases, we use the *split* strategy presented below.



a1 a2 b1 b2

Figure 3.2: Split procedure

We describe the splitting procedure using 2D illustrations. As the bounding cell ABCD approaches an obstacle from the left (See Figure 3.3a), we split the stream bubble vertically against Y axis. The natural split is along the obstacle's tangent lines of B'C' and A'D' (in fact they are tangent planes in 3D). The resulting bounding cells would then be AB'C'D and A'BCD'. Here the tangents are computed at the intersections of the bounding cell with the obstacle. In 3D, the tangent points are the maximum or minimum intersection points between the obstacle and the bounding volume. Note that for some cases, the maximum or minimum point might not just be on the edge as shown in the figure. At this time, we pick the one that follows the obstacle's shape best. This can be done by simply comparing sampled surface points of the obstacle in the moving direction X.

However, we have to know split direction first. There are four factors affecting the decision.

Splitting direction

- Moving direction: split will be perpendicular to the direction of motion. For example, in Figure 3.3b, the stream bubble (represented by its bounding box) is moving along the X direction. Therefore, the split will either be by Y (split into top and bottom halves) or by Z (split into left and right halves).
- Amount of overlap with obstacle: select the split direction with smaller overlap ratio. The ratio is calculated as the amount of intrusion by the stream bubble into the obstacle to the size of the stream bubble. This ratio is calculated in both possible directions, and the split is made along the direction with smaller intrusion. Using Figure 3.3c as an illustration, (ij/IJ) is the ratio along Z, while (hi/HI) is the ratio along Y.

- Shape of the obstacle: split is made along the direction of least resistance with the obstacle. Again using Figure 3.3c, the split direction is along the smaller of hi or hk.
- Use an arbitrary default split direction (Y in our case) when the three cases above fail to provide a decision.

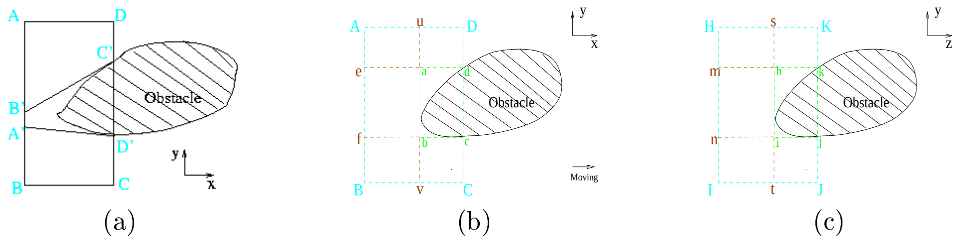


Figure 3.3: Determining split tangent plane and direction. ABCD and HIJK are 2D bounding boxes of a stream bubble cell. abcd and hijk are 2D bounding boxes of the intersection area between the obstacle and the stream bubble cell. In both images on the left, Z is pointing out of the page. On the right, X is pointing into the page.

Set new vertices

Splitting a stream bubble involves creating new vertices to go with each half of the stream bubble. That is, 8 new vertices need to be created. The general idea is to keep roughly half of the control points with each of the two new stream bubbles, while the other half will have to be created based on the split direction and the original control vertices. One possible way of carrying this task out is to vary the weights of the control points and introduce new vertices along the way.

In our approach, once we decide on the split direction, the split will proceed along the obstacle's tangent planes to maintain as much of the size and shape of the stream bubble. The setting of new points is similar to *erosion*. The difference is that the new set of points have to be within the region AB'C'D or A'BCD'. Figures 3.2 b1 and b2 show an example of how a stream bubble is split into two pieces.

3.2 Flows in Divergent or Vortex Regions

Let us look at the three cases in Figure 3.4. In all three instances, there are no obstacles. The rectangles represent the seed volume. One can see that as the rectangle gets deformed by the flow, it may stretch, curve, and generally distort. Since we are using bi-cubic surfaces to represent the bubble, it cannot represent a concave region along an edge. Hence, when the flow curvature is high, we need to split the bubbles so that they can better capture the local curvature of the flow. If the seed volume is simply stretching in opposite directions, the bubble will quickly fill the entire span of the volume making it appear static as opposed to moving through space. For this case, we provide the option of splitting such bubbles into smaller pieces so that they can continue to move and get distorted.

The splitting procedure for these cases is straight forward. We simple cut the bubble in half. This is implemented as a two step *break* procedure. We first check if the stream bubble's size is over some limit, and if so, we then check if the ratio of stream bubble cell's longest axis and shortest axis is over some limit. If both thresholds are exceeded, then a split along the center plane, whose normal is parallel to the longest axis, will happen. The setting of new added vertices is similar to *split*.

4 IMPLEMENTATION AND RESULTS

The following procedure is used to render stream bubbles in the 3D flow field.

1. Initialize seed volume(s). A seed volume can be specified either as one or more contiguous computational grid cells (assuming hexahedral cells) or by specifying bounds in physical space.

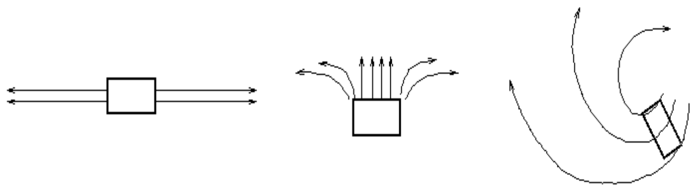


Figure 3.4: Divergent flows

2. Use the 2nd order Euler-Cauchy algorithm or the 4th order Runge-Kutta algorithm to calculate the next step positions of the 8 corner vertices.
3. If a vertex falls within a no data zone (e.g. inside an obstacle), apply the *erosion* procedure.
4. If there is a non-empty intersection between an obstacle and the cell or volume of the stream bubble control vertices, apply the *split* procedure.
5. If stream bubble stretch too much, then use *break* procedure.
6. If some vertices are outside the boundary, or the volume is getting too small, then terminate stream bubble.
7. Update stream bubble shape and position, go to step (2).

4.1 PLOT3D Data Set

We use synthetic data sets as well as CFD data sets from NASA to test our method. The data sets are in PLOT3D format [1] which includes (a) a file with the physical coordinates for each grid point in the computational grid, and (b) another file with field information calculated at each of the grid positions. Flow velocity is a derived quantity from these field information. Point location calculations is facilitated by the use of the Field Encapsulation Library (FEL) [3].

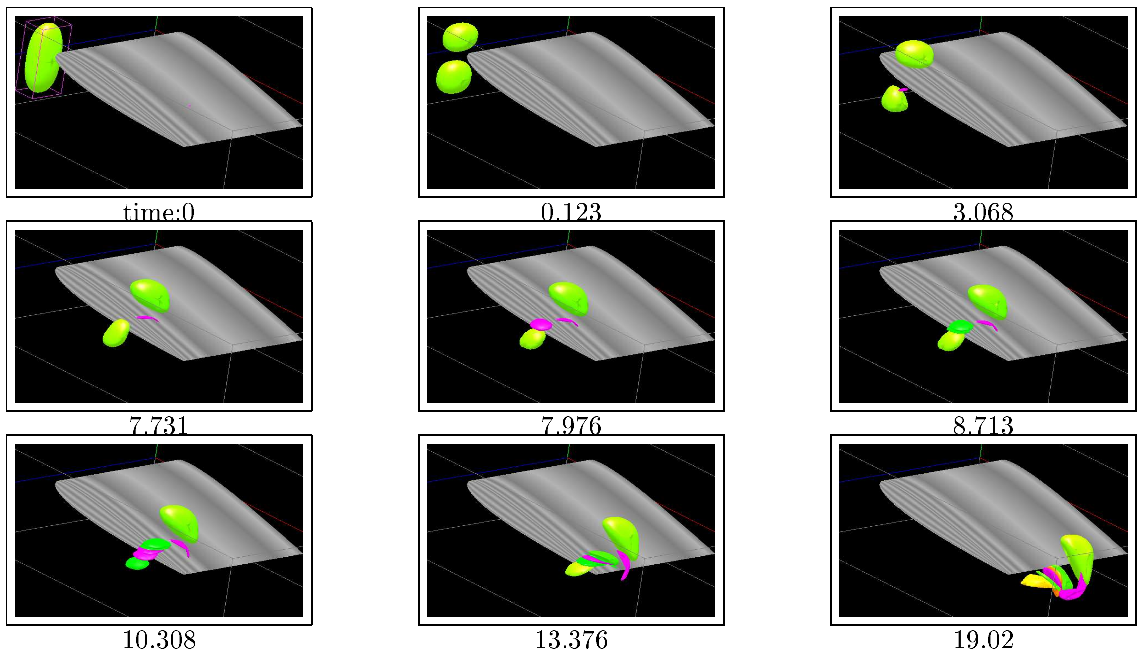


Figure 4.1: Time series of a stream bubble undergoing split and erosion. Color is mapped to relative volume change of the stream bubbles.

Figure 4.1 shows a time series of a stream bubble flowing around a wing. The center gray object is the geometry of a wing. In the beginning, the stream bubble undergoes a split process. Then the top half of the split stream bubble mainly rotates and deforms, and the bottom half slightly

5.

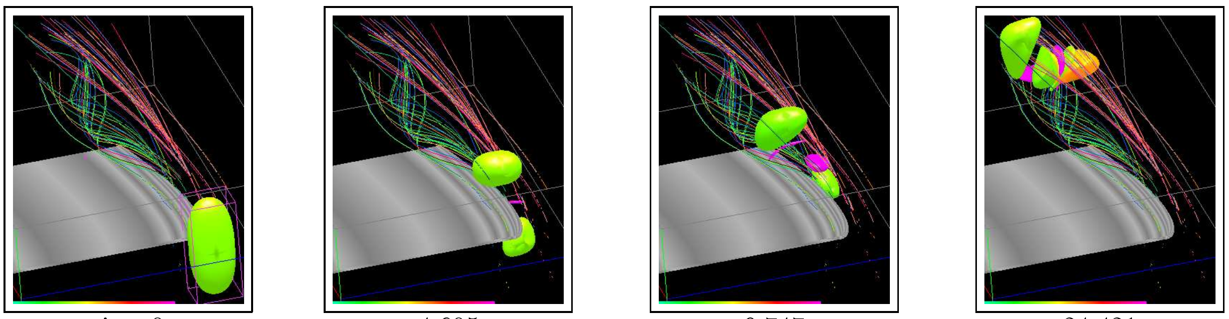


Figure 4.2: Streamlines super-imposed with the bubbles. Color is mapped to relative volume change of the stream bubbles.

erodes while it flows under the wing. Later, the bottom half has several splits, they twist and rotate more radically. The color of the stream bubbles are mapped to the relative volume change of the stream bubbles, while other color mappings such as absolute volume, velocity magnitude etc. are also available as options. Figure 4.2 is another view for Figure 4.1, and stream bubbles are drawn together with streamlines. The streamlines are seeded from the control points and are colored according to the control points' sequence (e.g. control point 0 is always colored green, etc.). The stream bubbles are generated at interactive rates. Local details can be seen at different stages through the flow. Shape deformations, stretching, rotation and volume changes can be observed as the stream bubbles move through the flow field.

4.2 Spiral Data Set

We also used a mathematically defined spiral data set generated by specifying rotation and translation parameters. There are no obstacles in the field. Looking at Figure 4.3, the spiral flow goes along from left to right along the Z axis which is centered in the middle of the flow volume. Here we present two sets of flow sequences using the same spiral data set.

The flow in Figure 4.3 is free. As the stream bubble rotate in the field, it also stretches along axis Z, and finally occupies almost the whole field. But it is not representative, because the local flow structure in fact is much more delicate. Figure 4.4 is front view from axis Z of Figure 4.3. It clearly shows the bubble is across the vortical area. In Figure 4.5, the stream bubble is modified by our *break* method. The thresholds for breaking is interactive. With break scale thresholds (size and axis ratios) set to nine and two, or when the volume of a stream bubble is over nine times of the predefined base volume, and when the ratio of the longest axis and the shortest axis of the stream bubble is over two, the stream bubble will split. In Figure 4.5, the stream bubble breaks into two in the second sequence, and break again into four in the third sequence and more in the following frames after the stream bubbles are stretched beyond the thresholds. Figure 4.6 is front view of Figure 4.5, but now stream bubbles' flow follows the vortical path.

For both Figure 4.4 and Figure 4.6, we use texture mapping which gives better depiction of local rotation and expansion. In Figure 4.3 and Figure 4.5, we use a blue-red (low to high) color mapping based on absolute volume magnitude. When stream bubbles stretch or expand, or when stream bubbles break, color will change correspondingly.

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 vortical field;

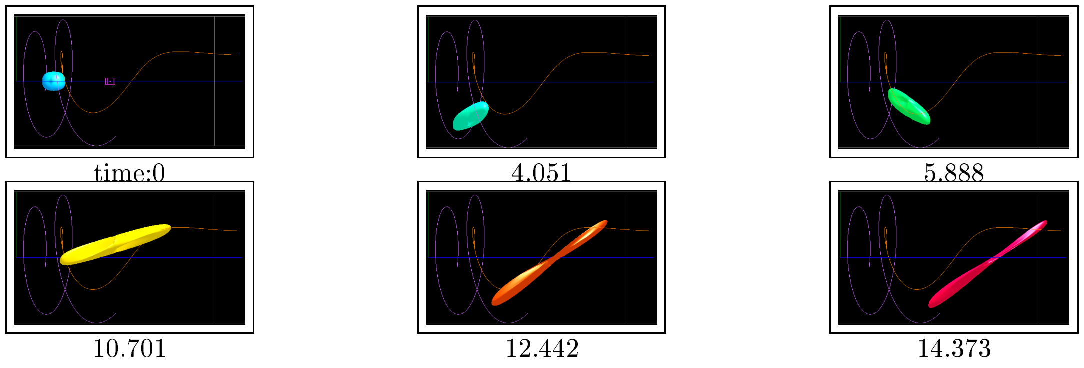


Figure 4.3: Side View: Time series of a stream bubble in vortex field without *breaks*. Color is mapped to volume of bubble. Note that bubble gets elongated but does not bend and curve as expected because of the degree of the NURBS representation used.

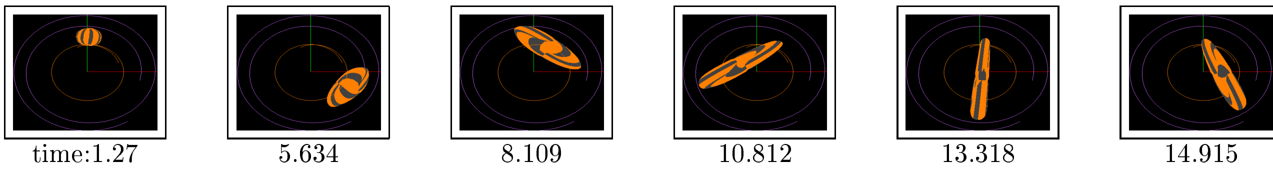


Figure 4.4: Front View: Time series of a stream bubble in vortex field without *breaks*. Same as previous figure but with texture mapping.

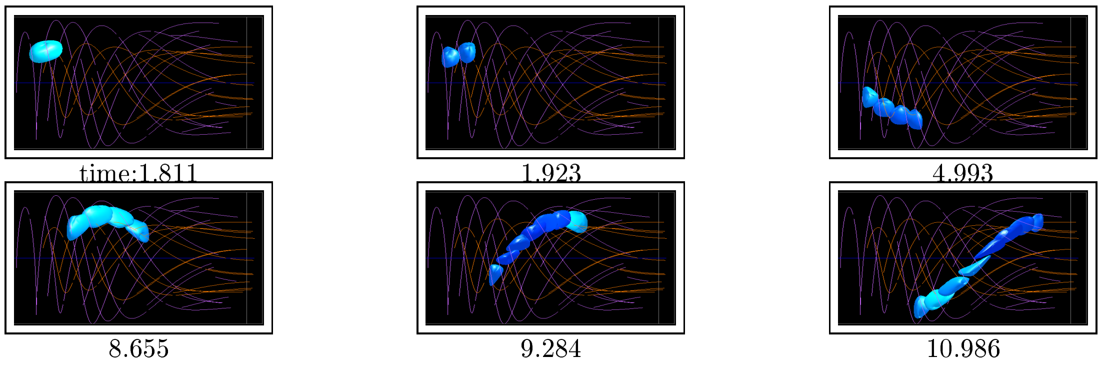


Figure 4.5: Side View: Time series of a broken stream bubble in vortex field. Color is mapped to bubble volume. This time the curvature of the flow is manifested in the chain of bubbles.

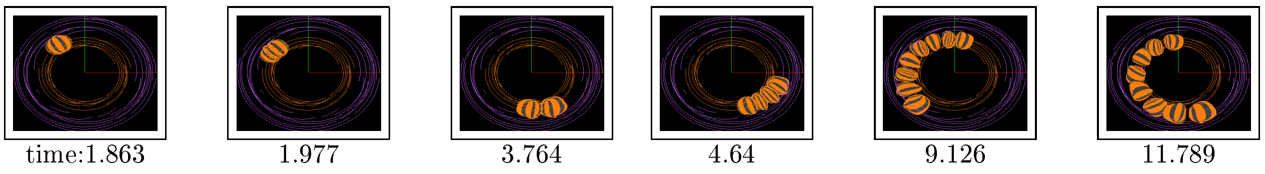


Figure 4.6: Front View: Time series of a broken stream bubble in vortex field. Same as previous figure but with texture mapping.

- 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 or breaking process; evaluating other types and degrees of surface representation aside from the bi-cubic NURBS used in this paper; extensions to non-hexahedral seeds so as to support flow data defined over unstructured meshes; and extension to time dependent flow data 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, NSF ACI-9908881, ONR N00014-00-1-0764, LLNL Agreement No. B347879 under DOE Contract No. W-7405-ENG-48, and DARPA grant N66001-97-8900.

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] Stephen T. Bryson, David Kenwright, and Michael Gerald-Yamasaki. FEL: The Field Encapsulation Library. In R.D. Bergeron and A.E. Kaufman, editors, *Proceedings of Visualization 96*, pages 241–247. ACM, October 1996.

[4] 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.

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

[8] 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.

[9] David F. Rogers and J. Alan Adams. *Mathematical Elements for Computer Graphics, 2nd edition.* McGraw-Hill, 1990.

[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] 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.

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

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

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

[16] Alan Watt. *3D Computer Graphics, 3rd edition.* Addison-Wesley, 2000.

[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.