Accelerated Streak Line Computation Using Adaptive Refinement

Alexander Wiebel
Max Planck Institute for Human Cognitive and Brain Sciences
Stephanstraße 1A
04103 Leipzig, Germany
wiebel@cbs.mpg.de

Qin Wang Dominic Schneider Gerik Scheuermann
Image and Signal Processing Group
University of Leipzig
PF 100920
04009 Leipzig, Germany
schopenkitto@hotmail.com, {schneider,scheuer}@informatik.uni-leipzig.de

ABSTRACT

We introduce an improved algorithm for streak line approximation in 2D and 3D time-dependent vector fields. The algorithm is mainly based on iterative refinement of an initial coarse approximation of the streak line. The refinement process is steered by a predicate indicating the local approximation quality. We apply the algorithm to several real-world data sets to prove its applicability and robustness. An error analysis and a comparison show that the algorithm produces more accurate streak lines in shorter time compared to previous approaches. Finally we show how the new algorithm can help to improve the construction of streak surfaces.

Keywords: Flow visualization, streak lines, adaptive refinement, unsteady flow, time-dependent vector fields.

1 INTRODUCTION

Unsteady flows are ubiquitous in nature and consequently in science. To be able to deduce the laws governing the behavior of flowing fluids scientist perform experiments and simulations. In both cases the effects under investigation appear to be *invisible* quite often. Making them visible is crucial and common. In experiments, a widely used technique is the injection of visible material into the fluid. Examples are dye for hydrodynamic and smoke for aerodynamic experiments. If a material is constantly injected from a fixed location, the patterns that become visible in the flow are called *streak lines*.

The same patterns are achievable for vector fields stemming from computational fluid dynamics (CFD), i.e. from flow simulations. The patterns are again called *streak lines*. Here a certain number of virtual particles is placed in the vector field at a fixed location at consecutive times and the particles are computationally traced through time and space [KL96, SMAM99]. The whole procedure will be described in more detail in the next section.

The aim of this paper is to significantly accelerate the computation of such streak lines in flow data given as 2D or 3D time-dependent vector fields. The central idea is to reduce the number of traced particles while keeping the accuracy of the streak line approximation as high as possible, i.e. to trace only particles at those parts of the streak line that experience strong deformation by complex patterns in the vector field.

2 STATE OF THE ART

The first part of this section outlines the mathematical basis for particle tracing and the techniques that were used for streak line approximation up to now. The second part summarizes work related to streak lines and their effective use for flow visualization.

Streak Line Computation

As already adumbrated, streak lines can be described as a continuum of particles that were injected into a flow at the same location but at different times. To enable their computational approximation we will give the mathematical definition of streak lines. To ease understanding of the definition, we first review the definitions of two other line types, streamlines and path line, too.

For the following definitions let

$$\mathbf{v} : \mathbb{R}^3 \times [t_{min}, t_{max}] \rightarrow \mathbb{R}^3$$

be a continuous time-dependent vector field. Let $\mathbf{a} \in \mathbb{R}^3$ be the position of a particle in space and let $t \in [t_{min}, t_{max}]$ be a certain time.

Stream Lines Stream lines are integral curves $\mathbf{c}_{\mathbf{a},t}(u)$ of vector fields which are tangential to the vectors of a field's domain. They can be interpreted as trajectories of particles in a steady flow. For time-dependent vector fields streamlines are of little use as they stay in a single time step. Thus they do not show actual particle motion but theoretical trajectories of particles with infinite velocity.

$$\begin{aligned} u &\mapsto \mathbf{c}_{\mathbf{a},t}(u) \\ \mathbf{c}_{\mathbf{a},t}(0) &= \mathbf{a} \\ \frac{\partial \mathbf{c}_{\mathbf{a},t}}{\partial u}(u) &= \mathbf{v}(\mathbf{c}_{\mathbf{a},t}(u),t) \end{aligned}$$

Here u is a time-independent parameter, t selects the time from which the vectors $\mathbf{v}$ are taken and $\mathbf{a}$ is the streamline seed at $u = 0$.

Path Lines In contrast to stream lines, path lines $\mathbf{p}_{\mathbf{a},s}$ in unsteady flow, indeed, are the paths of moving particles. Path lines are obtained by integration over space *and* time.

$$\begin{aligned} t &\mapsto \mathbf{p}_{\mathbf{a},s}(t) \\ \mathbf{p}_{\mathbf{a},s}(s) &= \mathbf{a} \\ \frac{\partial \mathbf{p}_{\mathbf{a},s}}{\partial t}(t) &= \mathbf{v}(\mathbf{p}_{\mathbf{a},s}(t),t) \end{aligned}$$

Here s is the seed time. Note that path lines and stream lines are identical for steady flow.

Streak Lines Streak lines $\mathbf{l}_{\mathbf{a},t}$ are imaginary lines connecting the locations of particles that were released into a flow at consecutive time steps. The lines can be observed when looking at the particles at a certain time t. As streak lines consist of particles and we need to trace these along their paths through the field, it is useful to describe streak lines in terms of path lines:

$$s \mapsto \mathbf{l}_{\mathbf{a},t}(s) = \mathbf{p}_{\mathbf{a},s}(t) \qquad (1)$$

Note that t is fixed and s varies. Like path lines, streak lines coincide with stream lines in the steady case.

It is worth mentioning that there exists another type of streak lines. In contrast to ordinary streak lines, the seed point for the particles varies in this new concept, i.e. instead of a constant position $\mathbf{a}$ they use a location $\mathbf{a}(s)$ moving along a path $\mathbf{q}$. These so called *generalized streak lines* $\mathbf{l}_{\mathbf{q},t}(s) = \mathbf{p}_{\mathbf{a}(s),s}(t)$ where introduced by Wiebel et al. [WTS$^+$07] just recently. Their paper shows the usefulness of generalized streak lines in the context of flow separation at boundaries of object immersed in the flow.

Related Work

A general overview of line based visualization of fields can be found in the State-of-the-Art report by McLoughlin et al. [MLP$^+$09]. In the following we will go into detail only for a number of streak line techniques especially relevant for our work.

An early implementations of streak lines for large unsteady flow fields was reported by Lane [Lan93]. In the literature we found only few papers dealing with accelerating streak lines. All of the reported approaches are complementary to what we present. One of the first ideas was trying to accelerate streak line computation by simply accelerating particle tracing. The first paper in this direction was written by Kenwright and Lane [KL96]. They increase the performance of the particle tracing by improving the cell search which is necessary for interpolation in unstructured grids. Later papers use also improved integration schemes, see [MLP$^+$09] for details.

A paper by Sanna et al. [SMA00] might seem similar to our work because they use *adaptive streak lines* for visualizing unsteady flows. However their method is quite different and does not have the aim to accelerate streak line computation. They are interested in a large number of densely seeded streak lines that produce visualizations similar to texture-based flow visualization [LHD$^+$04]. Their approach is only adaptive regarding the seeding of the streak lines which is steered by the local vorticity of the flow field.

Becker et al. [BLM95] extend the one dimensional streak lines to flow volumes that are seeded at several positions lying on a polygonal surface instead of only from one position. Particles are injected and traced from all positions simultaneously thus producing an evolving volume. They use an adaptive insertion of particles to keep the flow volume dense. Their criterion for insertion of new particles is the distance between particles. This is similar to what we present as distance criterion for streak lines. We will show that a distance based criterion is not the best approach to steer refinement.

Just recently flow visualization research has focused on streak surface visualization. Streak surfaces can be observed when particles are seeded from a line instead of a point. They correspond to a continuum of streak lines. The methods reach from a straight forward algorithm producing special streak surfaces called *eyelet path surfaces* [WS05], over GPU implementations for structured grids [vFWTS08, BFTW09], to a method that manages a complete triangulation of the surface and is also applicable to data given on unstructured grids [KGJ09].

Although many papers report the effectiveness of streak lines [Lan93, KL96, MLP$^+$09] for flow analysis others emphasize that care has to be taken when interpreting the resulting visualizations [Ham62, WH80, KS86].

3 ACCELERATED STREAK LINE COMPUTATION

Before we start to explain the actual technique, we should state that it is not needed for streak lines in steady flow. As streak lines are identical to streamlines in the steady case, the usual streamline approximation methods are sufficient. Also note, that the presented technique is applicable to generalized streak lines without any changes because both, ordinary and generalized streak lines, have the same digital representation as a polyline connecting the particles injected at different times.

Idea

The idea of the approach we present is to accelerate the computation of a streak line by reducing the number of particles traces that have to be computed. It is clear that we can only compute an approximation of the real streak line because we can trace only a finite number of particles in finite time. However, the aim still has to be as accurate as possible. Thus, we try to trace as few particles as are needed for a certain accuracy. We begin with a very coarse approximation, i.e. with only few particles started at equidistant points in time. Each traced particle results in a sample on the streak line (Fig. 2 left). The samples are joined with straight lines to form a polyline approximation of the streak line. Actually, the polyline is a linear interpolation of the samples. Please note that even though the particles all have the same distance in time, the samples do not have to have the same distance in space. In fact, most of the time this is not the case because the streak line is usually stretched and compressed.

To increase the accuracy of the approximation one decreases the time interval between the traced particles. The simplest and most common approach for this refinement is to decrease all time intervals uniformly, yielding an equidistant seeding in time again. This is most conveniently achieved by seeding one new particle in each interval between the particles of the coarser approximation. Thus is it possible to reuse the previously computed samples. The sampling density is doubled by this step.

The approach we suggest is to add new particles only in intervals in which the accuracy is low, i.e to steer the refinement by the local accuracy. The middle and right images in Figure 2 show the result of the two approaches applied to the coarse approximation shown in the left image of the same figure. Comparing the number of samples it becomes clear that the common approach is less efficient, as it produces more samples for a comparable accuracy.

Accuracy Estimation

In order to be able to seed particles only in intervals with low accuracy we need to measure the accuracy somehow. The best measure would be the distance between the approximated line and the correct streak line. As we do not have the correct streak line, we need to estimate the accuracy from the approximation alone.

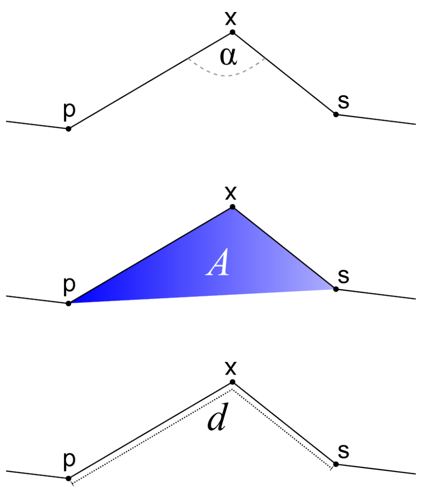

Figure 1: Different refinement criteria: angle α, area A and distance d.

Therefore we introduce three heuristic quality criteria in this section.

Figure 1 illustrates the three different quality criteria we employ for steering the refinement process. For each of the criteria we consider three consecutive points on the polyline representing the approximation: The current point x, its predecessor p and its successor s. Let $\mathbf{x}_p = p - x$ and $\mathbf{x}_s = s - x$.

Angle As the angle α between $\mathbf{x}_p$ and $\mathbf{x}_s$ comes closer to 180° the polyline becomes more similar to a straight line at x. As we use linear interpolation between the samples, a straighter polyline implies a higher accuracy. Thus we prescribe the desired quality using a minimum angle threshold k_α. The desired quality is reached if

$$k_\alpha < \arccos(\mathbf{x}_p \cdot \mathbf{x}_s)$$

with $\mathbf{x}_p$ and $\mathbf{x}_s$ being normalized vectors.

Area A small area A also indicates a relatively straight line. Thus we prescribe the desired quality using a maximum area threshold k_A. The desired quality is reached if

$$k_A > \frac{1}{2}\|\mathbf{x}_p \times \mathbf{x}_s\|.$$

Distance A small distance between samples naturally guarantees a high quality. Thus we prescribe the desired quality using a maximum distance threshold k_d. The desired quality is reached if

$$k_d > \|\mathbf{x}_p\| + \|\mathbf{x}_s\|.$$

We would like to mention, that one could make the above computations directly in homogeneous coordinates as suggested by Skala [Ska06]. It can be shown

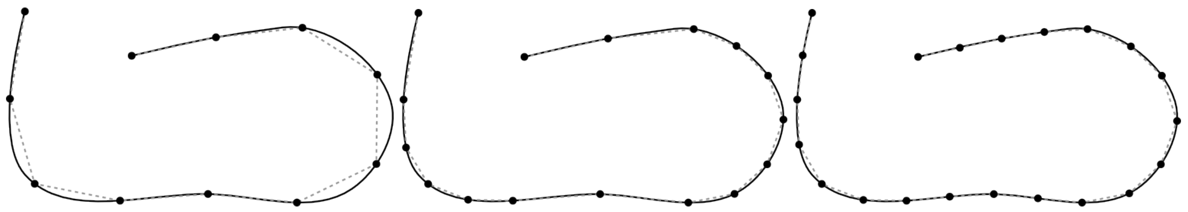

Figure 2: Comparison of refinement based on angle criterion and simple up-sampling. From left to right: Initial sampling of streak line, adaptive refinement and equidistant sampling with half of initial distance. For comparable precision adaptive refinement needs only 16 samples compared to 21 with simple sub-sampling.

that a solution of linear equations is equivalent to a generalized cross product.

Algorithm

In the following we describe the procedure for the adaptive streak line computation. As mentioned earlier, the first step of our algorithm constructs a coarse approximation of the desired streak line with the common method. This can be done quickly because the coarse approximation includes only the tracing of very few particles. Having the coarse line, we start with a first iteration along the line. We consider three consecutively computed particle positions and the intervals between them as a triplet. For each such triplet, starting with the one that has the youngest particles, we evaluate the quality criterion. If the criterion evaluates false we mark both intervals for refinement. After evaluation and marking have been performed for all triples we check all intervals. For each marked interval we trace a new particle starting at the time exactly in the middle of the times of the two particles bounding the interval. This completes the first iteration. We repeat the steps described for the first iteration until no intervals are marked during the evaluation phase or a maximum number of iterations has been reached. This completes the computation for the whole streak line. In the first case we have the streak line with desired accuracy ready. The second case is only for convenience. Particularly, it avoids infinite loops that might occur due user chosen thresholds that are beyond the sensible numerical bounds.

For the angle criterion we have implemented an additional condition that avoids endless refinement at sharp edges of the correct streak line. We omit the details of this condition here, as it will turn out later that the performance of the angle criterion is the worst. So we do not want encourage its reimplementation by the reader.

4 RESULTS

We applied our novel method to a number of data sets to prove its usefulness and robustness.

In this section we analyze two of these examples to demonstrate the efficiency of the new method and determine which of the quality criteria should be chosen. We compute streak lines using five different methods for both examples. At first we compute a reference streak line using the common method with a very high resolution. In all examples the reference streak lines have 5000 samples while the other streak lines compared against it consist of only up to several hundred particles. The reference streak line is considered to be the ground truth. To compare the efficiency of the different methods we compute a streak line with each of them. These streak lines start at the same position and cover the same time interval. For each method we vary the threshold for the used quality criterion and note the number of samples together with the achieved accuracy. The accuracy is measured by comparing the streak lines with the reference streak line and computing their distance. Our distance measure is evaluated as follows: For each point of the reference streak line we compute the shortest distance to the current streak line. Then we compute the mean of these distances. The resulting number is the average distance of the reference streak line to the current streak line and thus the error of the current streak line. We use the samples on the reference streak line to compute the distance and not those of the current streak line in order to have the same sampling density for all distance computations. This is important to make the computed distances comparable.

The four different streak lines we compare against the reference streak line are the three described above, i.e. angle-based, area-based and distance-based adaptive streak lines and a non-adaptive streak line that is equidistantly sampled with a prescribed number of samples. We carry out the comparison for three test cases, i.e. streak lines from one position in a 2D data set and streak lines from two different positions in a 3D data set. The 2D data set is the simulation of the flow around a cylinder (see Figure 3). The three-dimensional data set represents the fluid flow around a cuboid (see Figure 4). Figures 3 and 4 show the shapes and start positions of the considered streak lines.

The results of the analysis of the different streak line methods are shown as graphs in Figure 7. The graphs reveal that the angle criterion performs worst. The adaptive computation using the angle criterion is even worse than the non-adaptive method. The results of the distance-based method are not that clear. In the first two examples, where the streak line has nearly

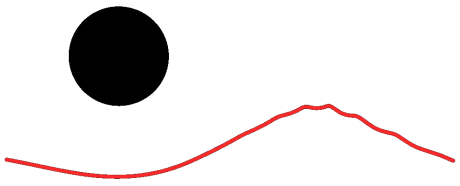
Figure 3: Streak line in flow around cylinder.

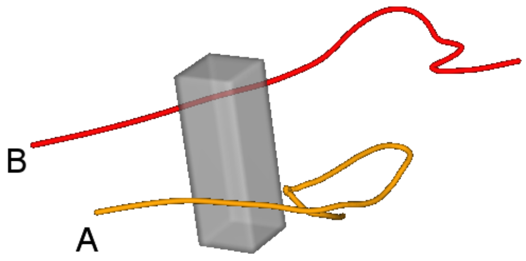

Figure 4: Streak lines in flow around cuboid. Labels A and B mark the seed points for the orange resp. red streak line.

no straight parts that could be exploited by the adaptiveness, the distance-based refinement is not as accurate as the non-adaptive streak lines (upper two images). If the real streak line, however, comprises some straight parts both, the non-adaptive and the distance-based adaptive method, perform very similar (lower image). The best method in all examples is the area-based adaptive method. It is only slightly better where the adaptiveness is not very useful (upper two images) but develops its full efficiency where the adaptiveness can sample straight parts sparsely (lower image). In the latter case the area-based method performs approximately three times better than the distance-based and the non-adaptive method (note the log scale of the graphs). In all our experiments the time used for particle tracing outweighs the time used for evaluating the quality criteria.

Finally, it should be noted that the number of computed samples increases monotonic with monotonic changes of the the quality criteria (increasing angle, decreasing area or decreasing distance).

Read-world Example

The cylinder and cuboid data sets discussed so far are of academic nature. We applied our new algorithm also on a real application data set: an aerodynamic simulation of a delta wing. The resulting streak lines can be seen in Figure 6. As the images show the methods works also on this complex data given on a large unstructured mesh consisting of 11 million cells (tetrahedra and prisms). As the acceleration method itself does not depend on the underlying mesh, it performs well as expected.

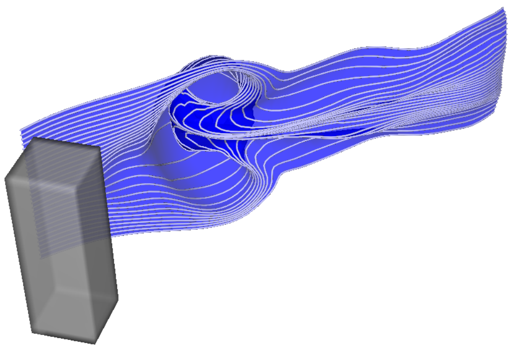
Figure 5: Thirty streak lines in flow around cuboid. Surface connecting the streak lines in blue.

Streak Surface

Our method is suitable to accelerate streak surface algorithms that build up the surface by a number of streak lines. By accelerating each streak line computation as described in this paper the whole computation time for a surface can be reduced. The streak surface shown in Figure 5 is constructed in this way, i.e. using a number of streak lines computed with the adaptive method and connected by a greedy tiling as described by Hultquist for his stream surface algorithm [Hul92].

Note, that the latest published algorithms for streak surface computation which we mentioned in the related work section chose different techniques. They do not construct streak lines explicitly but trace the particles of the streak surface independently.

5 CONCLUSION

In this paper we have presented a novel adaptive method to approximate streak lines. We have applied the method to data sets in 2D and 3D to demonstrate its usefulness. Additionally, we have carried out a thorough performance analysis that showed that only one of the three suggested quality criteria outperforms the conventional equidistant sampling. The analysis showed that the area-based method performs three times better than the common non-adaptive method.

We are aware that the initial coarse approximation might miss some of the smaller features of the correct streak line. The Nyquist-Shannon sampling theorem implies this restriction. However, smaller features are most often found in a later refinement step resulting from the evaluation of one of our quality criteria in its neighborhood.

For the future, we plan to enhance the accuracy between the sample points by replacing the linear interpolation producing polylines by higher order interpolation schemes that produce smooth lines.

6 ACKNOWLEDGMENTS

We wish to thank Markus Rütten from German Aerospace Center (DLR) in Göttingen for providing

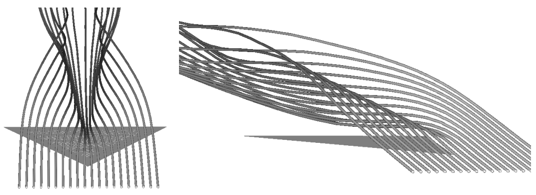

Figure 6: Streak lines in delta wing data set.

the cylinder dataset. The cuboid dataset results from direct numerical simulation carried out with the NaSt3DGP flow solver. NaSt3DGP was developed by the research group in the Division of Scientific Computing and Numerical Simulation at the University of Bonn. It is essentially based on the code described in a book by Griebel *et al.* [GDN98]. A version of the NaSt3DGP code, as well as related information and documentation is available for download at `http://wissrech.iam.uni-bonn.de/research/projects/NaSt3DGP/index.htm`. This work was partially supported by DFG grant SCHE 663/3-8.

REFERENCES

[BFTW09] Kai Bürger, Florian Ferstl, Holger Theisel, and Rüdiger Westermann. Interactive Streak Surface Visualization on the GPU. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2009)*, 15(6):to appear, November-December 2009.

[BLM95] Barry G. Becker, David A. Lane, and Nelson L. Max. Unsteady Flow Volumes. In Gregory M. Nielson and Deborah Silver, editors, *Proceedings of the 6th Conference on Visualization '95*, pages 329 – 335, Washington, DC, USA, 1995. IEEE Computer Society.

[GDN98] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, Philadelphia, 1998.

[Ham62] Francis R. Hama. Streaklines in a perturbed shear flow. *Physics of Fluids*, 5(6):644–650, June 1962.

[Hul92] Jeffrey P. M. Hultquist. Constructing Stream Surfaces in Steady 3D Vector Fields. In Arie E. Kaufman and Gregory M. Nielson, editors, *Proceedings of the 3rd conference on Visualization '92*, pages 171 – 178, Boston, MA, 1992.

[KGJ09] Hari Krishnan, Christoph Garth, and Kenneth I. Joy. Time and streak surfaces for flow visualization in large time-varying data sets. *Proceedings of IEEE Visualization '09*, October 2009.

[KL96] David N. Kenwright and David A. Lane. Interactive time-dependent particle tracing using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):120–129, 1996.

[KS86] M. Kurosaka and P. Sundaram. Illustrative examples of streaklines in unsteady vortices: Interpretational difficulties revisited. *Physics of Fluids*, 29(10):3474–3477, 1986.

[Lan93] David A. Lane. Visualization of Time-Dependent Flow Fields. In *Proceedings of the conference on Visualization '93*, pages 32 – 38, 1993.

[LHD+04] Robert S. Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23(2):203 – 221, 2004.

[MLP+09] Tony McLoughlin, Robert S. Laramee, Ronald Peikert, Frits H. Post, and Min Chen. Over Two Decades of Integration-Based, Geometric Flow Visualization. pages 73–92, Munich, Germany, 2009. Eurographics Association.

[Ska06] V. Skala. Length, area and volume computation in homogeneous coordinates. *International Journal of Image and Graphics*, 6(4):625–639, 2006.

[SMA00] Andrea Sanna, Bartolomeo Montrucchio, and R. Arina. Visualizing unsteady flows by adaptive streaklines. In *WSCG*, 2000.

[SMAM99] Andrea Sanna, Bartolomeo Montrucchio, R. Arina, and L. Massasso. A 3d fluid-flow visualizer for entry level computers. In *WSCG*, 1999.

[vFWTS08] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2008)*, 14(6):1396–1403, November - December 2008.

[WH80] David R. Williams and Francis R. Hama. Streaklines in a shear layer perturbed by two waves. *Physics of Fluids*, 23(3):442–447, 1980.

[WS05] Alexander Wiebel and Gerik Scheuermann. Eyelet Particle Tracing - Steady Visualization of Unsteady Flow. In Cláudio T. Silva, Eduard Gröller, and Holly Rushmeier, editors, *IEEE Visualization 2005 - (VIS'05)*, pages 607–614. IEEE Computer Society, October 2005.

[WTS+07] Alexander Wiebel, Xavier Tricoche, Dominic Schneider, Heike Jänicke, and Gerik Scheuermann. Generalized streak lines: Analysis and visualization of boundary induced vortices. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2007)*, 13(6):1735–1742, Nov.-Dec. 2007.

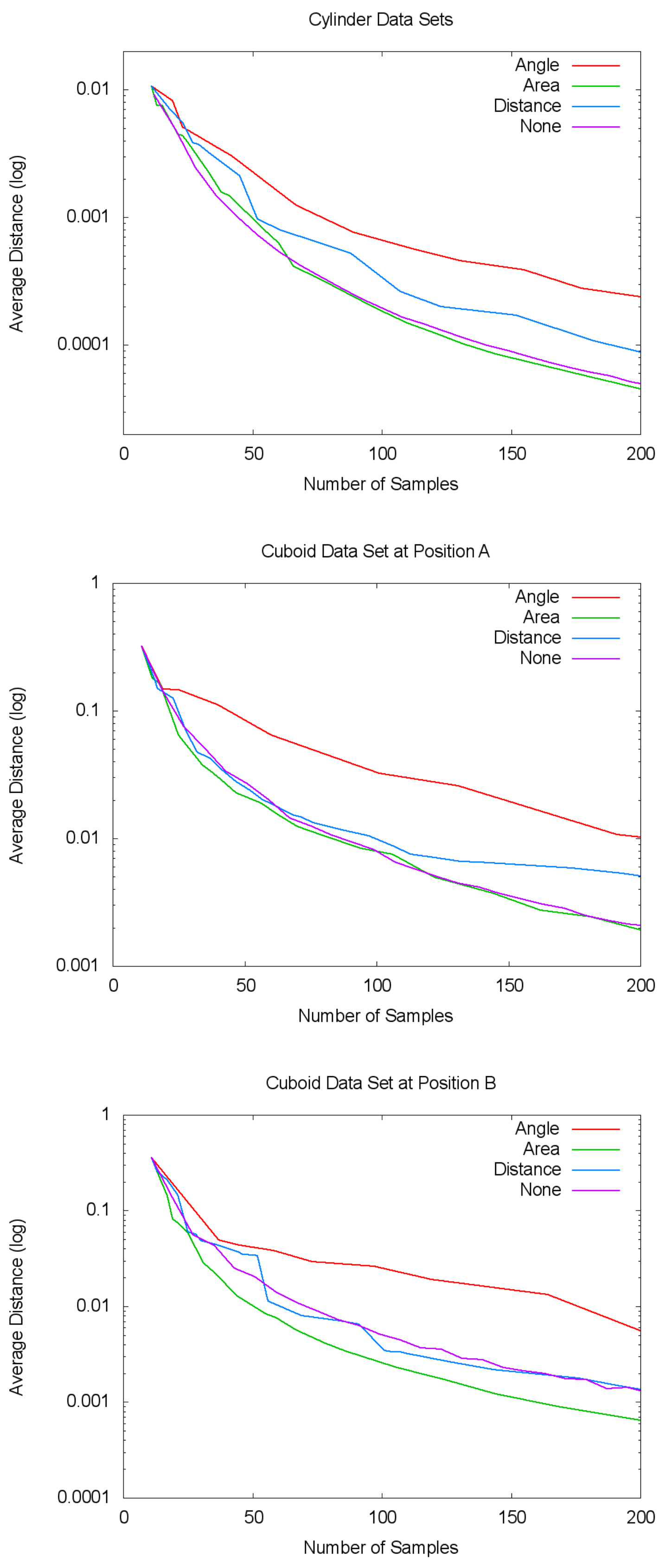

Figure 7: Error of non-adaptive streak line and adaptive streak lines using angle, area and distance criteria. Error is plotted versus number of samples. Streak line used as ground truth consists of 5000 samples.