LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-233865

Similarity-Guided Streamline Placement with Error Evaluation

Yuan Chen, Jonathan D. Cohen, Julian H. Krolik

August 21, 2007

IEEE Transactions on Visualization and Computer Graphics

Similarity-Guided Streamline Placement with Error Evaluation

Yuan Chen, *Student Member, IEEE*, Jonathan D. Cohen, and Julian H. Krolik

Abstract—Most streamline generation algorithms either provide a particular density of streamlines across the domain or explicitly detect features, such as critical points, and follow customized rules to emphasize those features. However, the former generally includes many redundant streamlines, and the latter requires Boolean decisions on which points are features (and may thus suffer from robustness problems for real-world data).

We take a new approach to adaptive streamline placement for steady vector fields in 2D and 3D. We define a metric for local similarity among streamlines and use this metric to grow streamlines from a dense set of candidate seed points. The metric considers not only Euclidean distance, but also a simple statistical measure of shape and directional similarity. Without explicit feature detection, our method produces streamlines that naturally accentuate regions of geometric interest.

In conjunction with this method, we also propose a quantitative error metric for evaluating a streamline representation based on how well it preserves the information from the original vector field. This error metric reconstructs a vector field from points on the streamline representation and computes a difference of the reconstruction from the original vector field.

Index Terms—Adaptive streamlines, vector field reconstruction, shape matching.

1 Introduction

Vector fields are commonly used to represent physical properties such as particle velocity or magnetic field across some domain. Visualization of vector fields is important for performing qualitative analysis in areas such as astronomy, aeronautics, meteorology, and medicine. The most common approaches are based on either line-integral convolution, which uses dense, highly local integration, or streamlines, which are longer integral curves with a more discrete, global flavor.

One particular area of scientific interest to us is the study of the turbulent behavior of ionized gases around a black hole. Velocity and magnetic field data are acquired using MHD (magnetohydrodynamics) simulation on a spherical domain. Because the domain of interest is a general 3D volume and is not restricted to a surface, sparse streamline approaches are preferable to textured approaches such as line-integral convolution.

As a sparse representation of a vector field, the quality of a set of streamlines is highly dependent on their *placement*, which includes a seed location and a length for each streamline.

If a specific type of feature of interest is known for the given application area and is mathematically well-defined, a *feature-guided* algorithm can tailor the placement to accentuate these features of interest. However, the discrete classification of features may suffer from robustness problems in practice.

When a feature-based approach is not applicable, a *density-guided* (or *distance-guided*) approach is typically employed. Density-guided approaches place streamlines to enforce a user-specified density function across the domain. The function may be constant, producing roughly evenly spaced streamlines, or it may be spatially varying (e.g., it may be mapped to vector magnitude or some other interesting scalar function). Notice that in these techniques, the density of streamlines is independent of the underlying flow function.

In this paper, we propose a *similarity-guided* placement strategy. We define a *similarity distance* as a metric for computation of local distance between streamlines. The similarity distance accounts not only for the closest distance from a point on one streamline to another streamline, but also accounts for their similarity of shape and direction. This is accomplished by examining the statistics of the Euclidean distance function as measured over a spatial window.

We employ this new similarity distance metric in the context of a simple, greedy algorithm for streamline generation. A large number of seed points are randomly ordered in a processing queue. Each seed point is grown using Runge-Kutta integration until its similarity distance from any streamline would fall below a prespecified similarity tolerance. This process may be performed in multiple passes to promote streamline fairness and to generate a multi-resolution streamline representation. As in some other density-guided approaches, the similarity tolerances may either be constant across the domain or spatially varying. We show that by using the similarity distance as a metric, even this simple placement strategy produces streamlines that naturally accentuate regions of geometric interest, with sparser streamlines in regions of more parallel flow.

In addition to visually demonstrating how the similarity-guided placement algorithm behaves on simple and complex vector fields in two and three dimensions, we introduce an error metric for streamline representations. This error metric is based on the idea that the streamlines are an alternative representation for the vector field, and should thus strive to preserve the information contained in the vector field. Our approach is to reconstruct a vector field from points on the streamlines and compute a difference of the reconstructed vector field from the original. We show that the similarity-guided placement strategy produces a significantly lower error for a given number of streamlines than a pure distance-guided strategy.

2 Related Work

Flow visualization has a rich literature, with a number of surveys available [4, 7]. We consider here the most relevant work in streamline visualization.

There are two competing goals in streamline visualization. One is to represent interesting features as well as possible. In principle, if we are allowed to place streamlines densely enough, we will not miss any information of the field. But practically, dense representations can cause severe clutter problems, especially in 3D. Thus the second goal is to have as little clutter as possible. Both feature-guided and density-guided placements try to find a good balance between these two goals.

Feature-guided strategies begin by identifying particular features in the field, that are of interest for a certain research or engineering problem, such as boundary layers, separation lines, separation bubbles, critical points, shock waves and so on. For many applications, the vicinity of critical points contains an interesting flow pattern. A critical point is defined as a point in the vector field where all components of the vector are zeros and the streamline slope is indeterminate. So no streamline will pass through it. For example, in a velocity field,

- *Yuan Chen is at Johns Hopkins University, E-mail: cheny@cs.jhu.edu.*
- *Jonathan D. Cohen is at Lawrence Livermore National Laboratory, E-mail: jcohen@llnl.gov.*
- *Julian Krolik is at Johns Hopkins University, E-mail: jhk@pha.jhu.edu.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

critical points are those with zero velocity magnitude. Critical points are further categorized into different types. The field is then segmented into regions, each containing a single critical point. Different seeding patterns may be used around the vicinity of the different critical points. Finally, some additional seed points are randomly distributed around the field.

Verma et al. [14] first proposed a feature-guided placement strategy for 2D vector field visualization. Ye et al. [15] extended this idea to 3D vector fields. They also provided a continuous mapping between seeding patterns and critical points, which accommodates the variability of critical points in 3D. In some applications, the number of critical points is small, so the number of streamlines generated by feature-guided strategies is likewise relatively small, and the clutter problem is minimal. But with a large number of critical points, the feature-guided approach won't necessarily produce a sparse streamline representation. Another drawback is the requirement of feature identification, which is not always available or easy.

Density-based strategies are usually associated with some density or distance metric. Turk et.al [13] proposed an image-guided placement strategy in 2D, which defined the density as the gray level of the low-pass filtered version of the image. Their optimization process adjusts the streamline placement to achieve a desired density by iteratively adding, growing, shrinking, merging, or removing streamlines.

Jobard and Lefer [5] proposed an evenly-spaced streamline placement algorithm in 2D where the density is defined by the Euclidean distance between adjacent streamlines. Mebarki et al. [10] proposed a farthest point seeding strategy for 2D vector fields. This greedy algorithm places one streamline at a time, always at the farthest point away from all existing streamlines to optimize continuity (i.e., it promotes long streamlines). They also used the Euclidean distance between adjacent streamlines as their density metric. Mattausch et al. [9] extended the idea of evenly spaced placement into 3D and used the 3D Euclidean distance as the metric. Liu et. al [16] introduced double queues for prioritizing topological seeding and favoring long streamlines. They also use cubic Hermite polynomial interpolation with RK4-ASSEC (fourth order Runge-Kutta integrator with adaptive step size and error control) to accelerate placement generation.

Li et al. [8] proposed a 3D image-space streamline placement method. They control the seeding and generation of streamlines in image space to avoid visual cluttering. This approach guarantees the minimum spacing between adjacent streamlines on the image plane and solves the clutter problem directly. But the quality of placement is highly dependent on a precomputed depth map, which provides the depth of 2D seeds in 3D object space. Schlemmer et al. [12] defined the streamline density of a region as the ratio between the number of occupied pixels by streamlines and the total number of pixels in the region.

3 Similarity Distance

3.1 Definition

The similarity distance is a window-based, point-to-streamline distance, where $d_{sim}(p, s_i, s_j)$ is the similarity distance from point p on streamline s_i to streamline s_j. For point p on streamline s_i, the window around p is a portion of streamline s_i centered at p with a length of w. The value of the similarity distance is computed as follows:

1. **Locate windows:** The first window in the comparison is centered at p. The other window is centered at q, the point on s_j with the smallest Euclidean distance to p. p and q are actual sample points generated by the adaptive Runge-Kutta integration process.

   If $i == j$, we are checking for *self-similarity*. In that case, we disqualify points that are too topologically close to p from consideration; all points under consideration for q must be at least d_{min} away from p as measured along the streamline. Thus the streamline i is only becoming self-similar if the window around p is similar to the window around some topologically distant part of s_i.

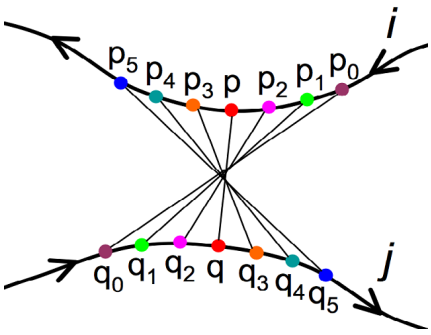


Fig. 1. Corresponding point pairs in two resampled streamline windows.

2. **Sample windows:** Center a window of size w about p (measured along the streamline), and uniformly resample using m ordered points, yielding $p_0, ..., p_{m-1}$. Similarly, uniformly resample over the window of size w about q using m ordered points, yielding $q_0, ..., q_{m-1}$.

3. **Compute similarity:** The result of the preceding resampling process is $m+1$ pairs of corresponding points. We use the distances between these corresponding points to compute the similarity between the two windows. Notice, as shown in Figure 1, that the orientation is significant. Thus two parallel streamlines are less similar if they the flow is going in opposite directions. We now compute the overall similarity distance from p to s_j:

$$d_{sim} = \|p-q\| + \alpha \frac{\sum_{k=0}^{m-1} \Big|\ \|p_k - q_k\| - \|p-q\|\ \Big|}{m} \qquad (1)$$

   The shape coefficient, α, is described below in Section 3.2.2.

3.2 Influence Factors

The two major terms of the similarity distance formulation Eq.(1), correspond to the two major influence factors respectively: (a) translational distance and (b) shape and orientation.

3.2.1 Translational Distance

The first term contributes the effects of Euclidean distance to the similarity distance. Given two streamlines s_i and s_j, translating them further from each other without otherwise changing their shape or orientation will monotonically increase the first term, while holding the second term constant. Thus all local similarity distances between the two streamlines will also increase monotonically as the translation increases.

3.2.2 Shape and Orientation

The second term contributes the effects of shape and orientation to the similarity distance. If the two streamline windows have identical shape and orientation, the second term is zero. As we deform parallel streamlines apart from each other without translating the window centers, the second term increases. Similarly, reversing the orientation of one of the streamlines increases the second term.

The second term measures the average deviation of point pair distances from the center point pair (translational) distance. In fact, if we replace the center point distance in the formula with the mean of point pair distances, and replace the L^1 norm accumulation with an L^2 norm, then the second term becomes the standard deviation of the point pair distances. In fact, we have experimented with using the mean and the standard deviation in the first and second terms, and achieved fairly similar results. However, we prefer the formulation presented above, because it gives special importance to the window center, which is the point at which we are growing a new streamline.

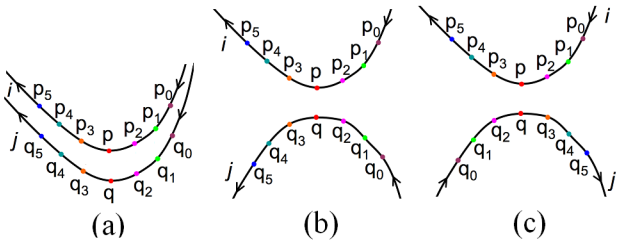


Fig. 2. The similarity distances from point p on streamline i to streamline j increase from left to right, due to change in shape and then orientation. The value of $\|p-q\|$ is the same for (a), (b) and (c).

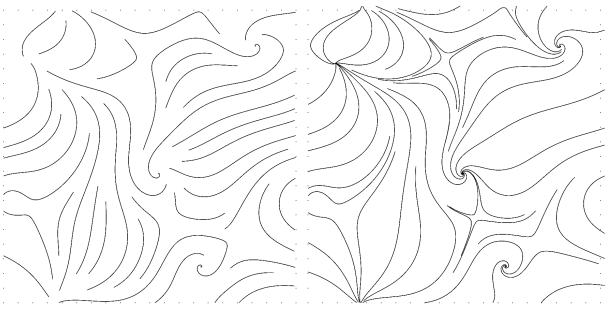


Fig. 3. **Left:** Euclidean distance metric (window size is zero), **Right:** Similarity distance metric (window size is six percent of domain width). Both images use 50 streamlines.

The shape coefficient, α, determines the relative importance of the two influence factors. In our examples, we have found values between 1 and 3 to be the most useful, with little noticeable change as we increase beyond that (this may have some interesting significance if you consider the second term to be a standard deviation). Notice that setting either α or the window size to zero results in a pure, Euclidean distance between p and s_j.

Figure 2 illustrates how the similarity distance is affected by shape and orientation. 2(a) through 2(c) with the same Euclidean distance between center point pair (p,q) are in order of increasing similarity distance. In 2(a), streamline i and j have similar shape and orientation. In 2(b), the shapes are quite different, but they both point from right to left. In 2(c), the shapes are the same as 2(b), but the direction of j is reversed, resulting in a larger similarity distance.

4 Streamline Placement

We can use the similarity distance metric from Section 3 to place streamlines over a domain. As in other distance-based streamline placement algorithms, the user chooses a distance of separation, d_{sep}, to enforce between streamlines. In the case of our similarity metric, d_{sep} will be the minimum distance between parallel streamlines. Where the streamlines are not parallel, the minimum separation will be reduced by some amount as controlled by setting the shape coefficient, α.

Figure 3 shows a comparison of streamlines generated using a Euclidean distance metric and our similarity distance metric. Both images contain 50 streamlines, but the left image uses a window size of zero, and the right image uses a window size that is six percent of the domain width (d_{sep} for the left image is set lower to achieve the same streamline count). The right image uses the 50 streamlines to more effectively portray the geometrically interesting regions, such as those near the critical points. Although there may be some natural inclination to judge the left image as prettier because it avoids the darker regions of increased density, the right image clearly conveys more information about the flow about the critical points. Figure 4 shows a similar comparison in 3D.

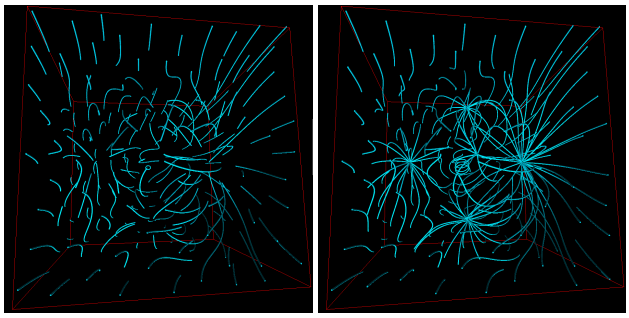


Fig. 4. **Left:** Euclidean distance metric, **Right:** Similarity distance metric in 3D using the same number of streamlines.

```
Placement(vecField, numSeeds, dSep, dSelfSep, minLen)
    Seeds = randomSeeds(vecField, numSeeds)
    while(Seeds is not empty)
        seed = Seeds.dequeue()
        Integrate(forward, vecField, seed, Lines, line)
        Integrate(backward, vecField, seed, Lines, line)
        if (Length(line) > minLen)
            Lines.insert(line)

Integrate(direction, vecField, seed, Lines, line)
    while(RK_OneStep(direction, line, p) succeeds)
        if dSim(p, line, line) < dSelfSep
            Close(p, line)  // optional
            return
        forall prevLine in Lines
            if d(p, line, prevLine) < dSep
                return
```

Fig. 5. The pseudocode for the basic streamline placement algorithm using the similarity distance.

4.1 Algorithm

To test the use of similarity as a metric for placing streamlines, we have developed a simple, greedy algorithm for streamline growth from seed points.

The pseudocode for our basic placement algorithm is shown in Figure 5. Placement begins by inserting a densely-sampled set of seed points into the *Seeds* queue in a random order. For each point in the queue, we iteratively grow the streamline by applying adaptive fourth order Runge-Kutta integration. As we grow the streamline, we check its similarity distance to all the previously placed streamlines. When the similarity distance falls below the user specified threshold, d_{Sep}, or if we reach a boundary of the volume or if integration becomes undefined due to a nearby critical point, we terminate the growth of that streamline. In the case of self separation (comparing a streamline to itself), we generally use a significantly smaller separation distance, $d_{SelfSep}$, to allow for the case of streamlines that form a closed loop. If the resulting streamline is longer than the desired minimum length, it is added to *Lines*, the set of placed streamlines. Otherwise, it is discarded.

In our implementation, we accelerate the above algorithm by placing all the sample points of the streamlines into a uniform grid, where the cell size is proportional to the separation distance. Because the similarity distance between point p on streamline s_i and point q on streamline s_j is always greater than or equal to the Euclidean distance between p and q, there is no need to find the closest point on streamlines that are farther from p than the separation distance, nor do we need to compute the actual similarity distance in that case. Of course, more sophisticated space partitioning structures may be used in place of the uniform grid, which has been sufficient for our test examples.

Although this simple placement algorithm seems to work reasonably well, it does have some shortcomings. It is unfair to the streamlines, favoring those with seeds early in the random queue over those later in the queue. The earlier seeds have the opportunity to grow longer, although there may be nothing particularly special about them. One could order the seeds according to some importance function, but often one cannot predict the importance of a streamline simply from its seed position.

One way to mitigate this situation is to grow the streamlines in several passes, with a schedule of decreasing separation distances. This is more fair to all streamlines, but does tend to produce shorter streamlines overall.

We believe it may be possible to achieve even better results by incorporating the similarity distance into a richer optimization framework with more legal moves for exploring the space of placements [13]. This is beyond the scope of this paper, but seems quite promising for the future.

4.2 Parameters

At first it seems that there are a large number of parameters to be set in our method. However, there is some logical interdependence among these that allows most of them to be set automatically in most cases.

- d_{sep}: This is the primary parameter for adjustment, which we set as a fraction of the domain width. It varies from 2.0% to 10.0% of the domain width in our examples shown in the paper.
- α: The most useful range for the shape coefficient is between 1 and 3. We typically get reasonable results by setting this between 2 and 3.
- w: The window width is also set as a fraction of the domain width. We usually set w in the range of 1 to 5 times of d_{sep}.
- $d_{selfsep}$: The self-separation distance is used for terminating or closing a streamline due to self-proximity. This should be smaller than d_{sep}, and we generally set it to $d_{sep}/10$.
- d_{min}: The minimum streamline length away from the central point for testing self-separation should be greater than $d_{selfsep}$ to prevent the two windows from having common samples (because of topological overlap). If it is too large, it will force small loops to go around multiple times before closing. We generally set this to be 3 to 10 times of $d_{selfsep}$.
- *minLen*: The minimum streamline length is largely a matter of taste. Keeping more of the short streamlines will prevent some potentially long streamlines from growing. It can be set as a multiple of the w, d_{sep}, or the domain width. We typically set it to $w*2$.
- *numSeeds*: In all of our tests, we use every sample in the vector field as a seed. It is possible to use more or fewer seeds if you know that the vector field is relatively over- or under-sampled with respect to the features it contains.

We usually define the domain width as the smallest one among all dimensions.

4.3 Favoring Interesting Features

In addition to Euclidean distance, the similarity distance measures the difference in shape and orientation between streamlines. Thus it naturally favors interesting regions, such as the vicinity of critical points and separating planes.

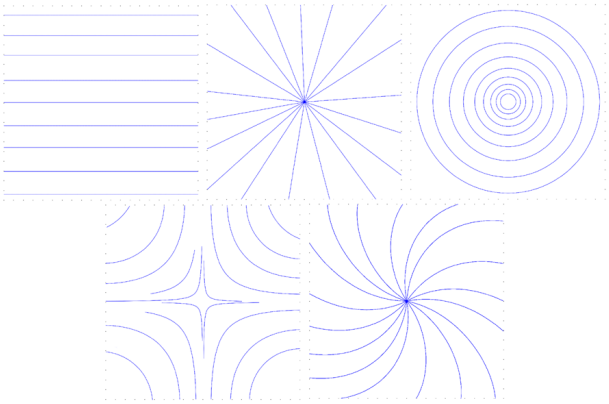


Fig. 6. The similarity distances used to generate these five images are the same. In the upper left image, the vector field is flat. The direction of vector data is horizontal at any position. The other four images all have exact one critical point at the center of field. The types of critical points are sink/source, center, saddle and spiral, respectively.

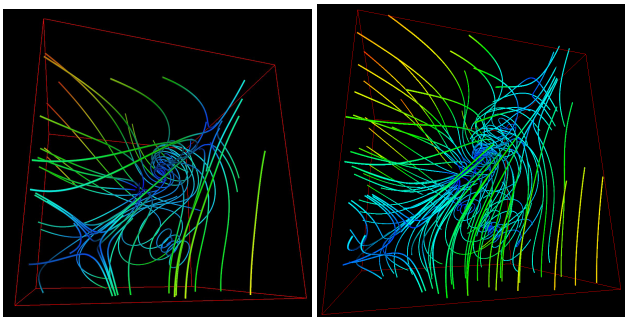


Fig. 7. On the left is a streamline visualization with 50 streamlines produced by [15] using explicit critical point detection and case analysis. On the right is a streamline visualization with 111 streamlines produced using our similarity-based placement technique on the same vector field data.

4.3.1 Critical Points

Many applications consider the critical points of vector fields to be interesting. As such, many feature-guided streamline placement strategies[14, 15] explicitly choose streamlines which emphasize the vicinity of these critical points. In Figure 6, we show in 2D that the distance between streamlines is naturally reduced around critical points as compared to parallel flows (they may even visually touch when rendered onto a discrete pixel grid). This is because the second term of Eq. 1 grows when the streamline integration approaches a sink or source.

We can also observe this behavior on fields with multiple critical points in 2D and 3D (see Figures 3 and 4). In fact, we can see in Figure 7 that similarity-guided placement can produce results somewhat comparable to those of [15], but without requiring explicit detection of critical points or special case analysis based on type of critical point. Note, however, that our algorithm does not guarantee emphasis of *all* critical points, because it uses no such feature detection. Whether the streamlines will miss critical points is dependent on the parameter values used as well as the spacing between critical points in the data. We can capture all the critical points in Figure 7 with comparable clarity, but it does require more streamlines than the feature-specific approach.

4.3.2 Separation

Separation is frequently studied in many fields, such as fluid dynamic or aeronautics. However, it is harder to define in closed form than

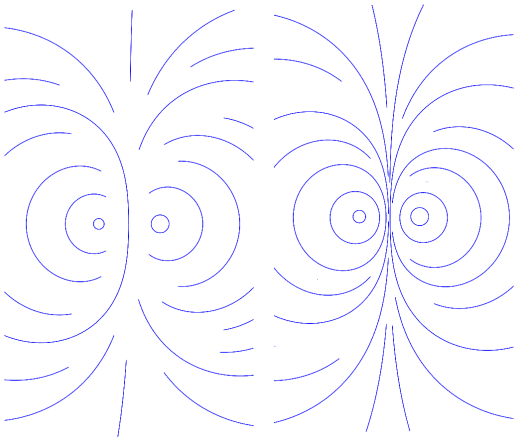


Fig. 8. Both of these two images contain 22 streamlines. The left one is generated using Euclidean distance based placement. The right one is generated using our method, which emphasizes the separation curve between the two critical points.

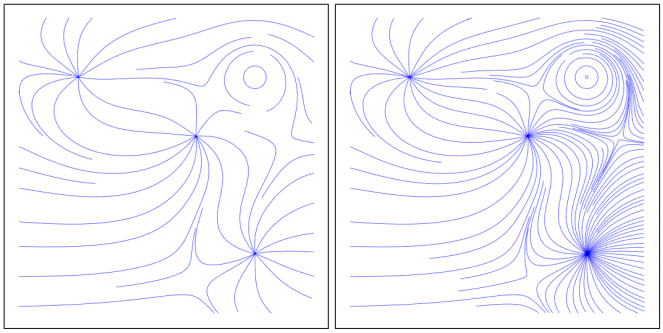


Fig. 9. **Left:** Streamlines generated using a uniform separation distance. **Right:** Streamlines generated using variable separation distance, decreasing linearly left to right.

a critical point, and it has higher dimension. Separating curves and surfaces are global features which no streamline will cross. The separation is either a closed curve/surface or intersects the boundary of the domain.

Our similarity distance-based placement will emphasize these separations similarly to emphasizing critical points. Figure 8 shows two sets of streamlines generated by a typical Euclidean distance method and by our method. Our method allows one to see the presence of a separating curve much more clearly.

4.4 Variable Density

In our method, the similarity distance is defined locally. Thus it is possible to specify the separation distance, d_{sep}, as a function of position in the domain rather than as a constant. This allows us to achieve variable density of streamlines. The separation distance is in this case a scalar field over the domain. It can be specified either a function to be evaluated or in sampled form (i.e., a texture image over the domain). It can be chosen manually to highlight some particular spatial region of interest, or it can be derived from the vector field or one of its associated scalar fields. Figure 9 shows a simple example of this effect, where the separation distance decreases linearly from the left to the right side of the domain.

4.5 Multiresolution

Exploring a complex vector field may require multiresolution streamline placement. Generally, a rough representation on the whole domain

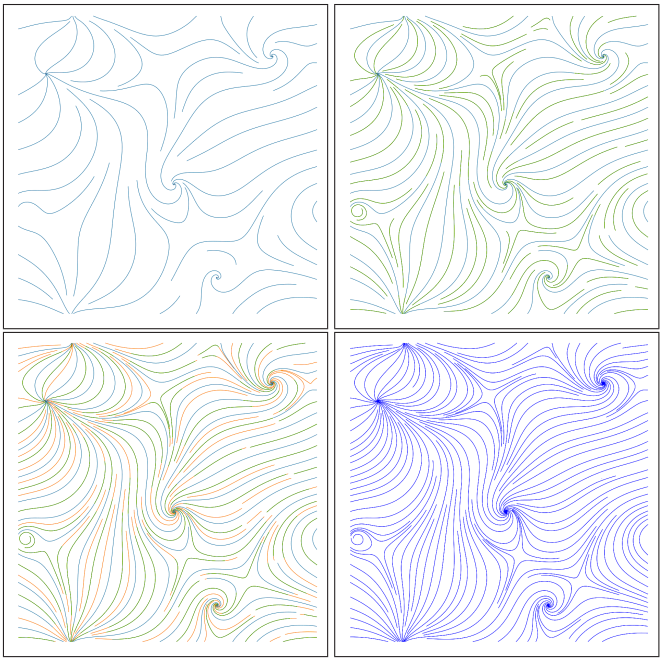


Fig. 10. Three levels of detail for a vector field, generated by successively decreasing d_{sep} from 6.0%, to 4.0%, to 2.0% of the domain width. For comparison, also see on the lower right the result of applying the smallest value of d_{sep} in a single pass, producing fewer short streamlines. For all four images, the window size is 10% of the domain width and the shape coefficient α is 3.0.

is provided at the beginning. Then the user may want to define the region of interest and have more detailed visualization there. Jobard and Lefer [6] proposed a nested streamline placements with increasing density. Their method doesn't give the streamline placed at a given level the chance to grow further in the finer levels. So it does have the limitation on producing long streamlines.In the method proposed by Mebarki et.al[10], they favor long streamlines by elongating all previously placed streamlines before each new placement of streamlines at all levels.

The multi-pass version of our placement algorithm, described above in Section 4.1, can be used as the basis for a simple multi-resolution streamline representation. Each pass successively reduces the separation distance, producing increasing levels of detail. By storing with each streamline the range of samples to be used for each level of detail, we can render at any of these levels of detail on demand. The result of this can be seen in Figure 10. We see three increasing levels of detail. With each new level, previously existing streamlines may grow, and new streamlines may be added. For comparison, we also see the result of generating the smallest separation distance streamline set using a single pass.

In our method, the previously placed streamlines and new seed points have a fairly similar chance to grow at each level.

5 Error Function

The goal of scientific visualization is to use graphical tools to reveal underlying properties of the data field in an easy-to-understand way. Given a set of relatively sparse streamlines, our brain may perform some form of interpolation to understand the blank areas between streamlines. So ideally, the result of such interpolation should agree with the underlying field data. In other words, if the visualization successfully represents most of information contained in the field, then the field data could be fully reconstructed from the visualization. Inspired by this, we define an error function for streamline-based vector field visualization.

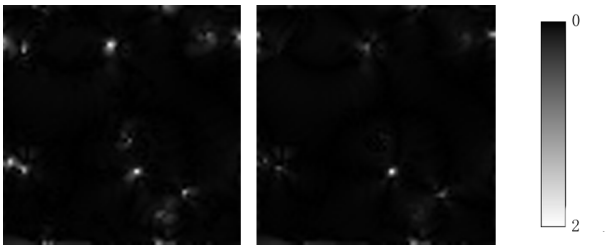


Fig. 11. The error values for the two sets of streamlines in Figure 3.

5.1 Definition

Given a vector field, V, containing n grid points, $V_0..V_{n-1}$, let S denote a set of sampled streamlines. If $V'(S)$ is the vector field reconstructed using S and resampled at the original n grid points, then we define the error field as the residual magnitude:

$$E_j(S) = \left\| V_j - V'_j(S) \right\| \tag{2}$$

and the total error as:

$$E(S) = \frac{1}{n} \sum_{j=0}^{n-1} E_j(S) \tag{3}$$

We intentionally use the residual in our formulation rather than the angle between the vectors because it is more general – it applies even if the vectors are not unit length. This brings up the interesting point that a basic streamline representation can only represent the directions of V, but not the magnitudes. In this case, we take the residual between normalized versions of V and $V'(S)$. However, in some cases, streamline representations may logically include the magnitudes at each sample. This makes sense, for example, if the magnitudes are used to color the streamlines. In that case, we can reconstruct a non-unit-length $V'(S)$ and take the residual between the two non-unit-length vector fields.

5.2 Vector Field Reconstruction

Reconstruction of a vector field from streamline sample points is a scattered data reconstruction problem. Although more sophisticated solutions to this problem are known (e.g., [2, 3, 11]), we use a linear interpolation approach as a proof of concept.

We first compute the Delaunay triangulation of the streamline samples in 2D or 3D (we use Qhull [1] in our current implementation). We currently use only unconstrained triangulation because the constrained Delaunay triangulation in 3D may be ill-posed, requiring the addition of extra samples (i.e., Steiner points). Next, for every sample point, we compute a tangent vector using local differences and normalize it.

To perform the reconstruction and resampling of the vector field, we perform a point location query in the triangulation for each grid point, then perform linear interpolation of the cell corners using barycentric coordinates and normalize the result. In the case of streamlines with magnitude values, it may be desirable to interpolate the vectors and the magnitudes separately. For points outside the convex hull of the sample points, we clamp to the nearest value on the convex hull as a simple extrapolation technique. We currently accelerate the point location queries using a uniform grid.

Figure 11 shows the residual magnitude fields for Euclidean distance-based placement versus similarity-based placement with the same number of streamlines (using the streamlines from Figure 3). It is not surprising that most noticeable errors occur near the critical points, and these tend to be reduced by our approach.

6 Results

So far, we have shown results on fairly simple, synthetic data as a way of validating our approach. Now we demonstrate on two more complicated data sets from real scientific simulations, then report some data from all the models.

Data	1.vec	om08	3d.vec	KDP
Struct	uniform	structured	uniform	curvilinear
Dim	70x70	251x159	128x128x128	192x192x64
d_{sep}	0.03	0.035	0.06	0.1
window	0.1	0.1	0.1	0.2
α	2.0	2.0	3.0	3.0
Time	13.2	122.5	558.9	974.5
Lines	154	166	126	148

Table 1. Statistics of several data sets. The d_{sep} is the distance of separation in terms of the percentage of field width, as is the window size. The processing time is in seconds.

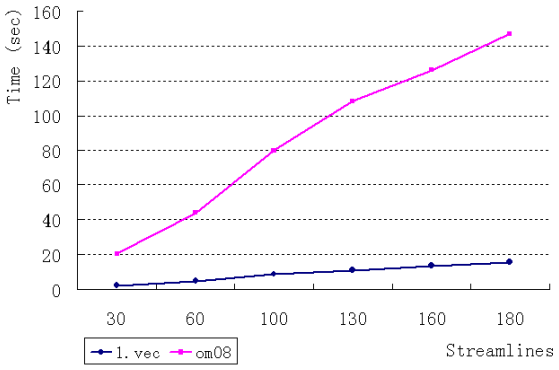


Fig. 12. Number of streamlines versus generation time for two data sets.

Figure 14 shows a 2D turbulent flow from a simulation of a swirling jet. It simulates the situation with an inflow into a steady medium. The domain is a structured grid with 251x159 cells. The topology of the flow has a very complicated structure and contains about 300 critical points. The extremely dense distribution of critical points makes it hard to seed streamlines using feature-guided strategies. Figure 14(a) shows the result using the typical Euclidean distance placement without detecting self closure. It contains some obvious visual clutter artifacts. The computational time is roughly three times longer than the others in Figure 14 because those closed streamlines will only stop integration when the integration length is beyond some user-defined maximum length, consuming unnecessary computational time. In Figure 14(b), we still use the Euclidean distance, but terminate the integration of closed curves. Figure 14(c) shows the result from our similarity-guided strategy with self closure detection. The separation distance is reduced from (a) and (b) so that we get the same number of streamlines in all three. Figure 14(c) favors interesting regions with longer and more streamlines. We can see more structure in (c) than in (b) almost anywhere we look closely. For comparison, Fig-

Data	Metric	Streamlines	Avg. Error	Improvement
om08	Euclidean	171	0.054	
	similarity	166	0.034	37%
3d.vec	Euclidean	127	0.12	
	similarity	126	0.084	30%

Table 2. Comparison of our method with Euclidean distance based method in terms of reconstruction error. The average error is measured by considering the vector data as all unit vector. The error would fall in the range of 0 to 2. The improvement gives the percentage of error reduction by using similarity distance instead of Euclidean distance

ure 14(d) uses the Euclidean distance based method with the same distance threshold of Figure 14(c), so it uses fewer streamlines overall, and captures even less structure than (b) or (c).

Figure 13 shows a 3D magnetic field from an MHD simulation of ionized gases around a black hole. The field is symmetric, so the streamlines may be traced through a spherical volume, though the data is in one sphere quadrant (minus some small cutouts at the center and down the polar axis). The curvilinear grid has a huge range of cell sizes, with the smallest ones at the smallest radii and closest to the equatorial plane. The magnetic field has no critical points. In Figure 13(a), we use a variable density function to emphasize the region close to the polar axis. Streamlines in this region tend to follow a fairly tight coil up the axis. In Figure 13(b), we emphasize the streamlines near the equatorial plane, which tend to wind around in a much broader pattern.

We have implemented the streamline generation algorithm on a Linux PC equipped with 2.4 GHz dual AMD Opteron 250 CPU, 8GB RAM. Table 1 presents some timing data for several test models. We use one seed point per grid sample. The 1.vec data (Figure 10) is a simple 2D synthetic data set provided with Abdelkrim Mebarki's streamline placement package. The om08 (Figure 14 is a complex simulation of a swirling jet with an inflow into a steady medium. The 3d.vec (Figure 4) is a synthetic data set we generated to contain six critical points. KdPhrg (Figure 13) is a magnetic field resulting from a relativistic MHD simulation of ionized gas in proximity to a black hole. Figure 12 demonstrates the relation between the processing time and the number of streamlines generated. Since the similarity distance check is the most crucial operation in our algorithm, the computing time grows fairly linearly with the number of generated streamlines. There is also a small factor corresponding to the number of seed points.

Table 2 shows the reconstruction errors of our method and Euclidean distance based method on 2D and 3D data sets.

7 Conclusion and Future Work

We have proposed the idea of a similarity metric for measuring streamline proximity, and demonstrated a streamline placement algorithm using similarity to drive the selection of streamlines and their lengths. This new approach favors streamlines near interesting flow features, such as critical points and separations, without the need to explicitly enumerate these features.

We have also introduced an error metric for streamline representations to measure how well they preserve the information in the original vector field. Using the metric, we can see that similarity-guided streamline placement outperforms Euclidean distance-guided placement not only visually, but quantitatively as well.

The placement algorithm still has some room for improvement. It may benefit from more carefully chosen order of seed point growth, such as the farthest point formulation of [10]. It may also be beneficial to add more types of moves to our optimization process, as in [13], making the initial placement order less crucial and less sensitive to a particular randomization. A more efficient spatial search structure can be used to improve the speed of generating the streamlines.

We choose to use a very intuitive and simple reconstruction method for vector field reconstruction in our error measurement, which consists of Delaunay triangulation and linear interpolation. It would be interesting to see some other reconstruction methods, which may be more suitable for this application.

Acknowledgements

We would like to thank Alex Pang and Xiaohong Ye for providing, the 5-critical-point 3D vector data, associated color function and image output from [15], and also Gerik Scheuermann, Xavier Tricoche, and Wolfgang Kollmann for providing the 2D swirling jet data. We would also like to thank Abdelkrim Mebarki for sharing his streamline placement package on line as well as some 2D vector field data. This work was supported in part by NSF ITR Grant AST-0313031.

References

[1] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, Dec. 1996.

[2] M. Bertram, X. Tricoche, and H. Hagen. Adaptive smooth scattered data approximation for large-scale terrain visualization. In *Proceedings of the Symposium on Data Visualisation 2003*, pages 177–184, 2003.

[3] J. Haber, F. Zeilfelder, O. Davydov, and H. P. Seidel. Smooth approximation and rendering of large scattered data sets. In *Proceedings IEEE Visualization 2001*, pages 341–348, 2001.

[4] H. Hauser, R. S. Laramee, and H. Doleisch. State-of-the-Art Report 2002 in Flow Visualization. Technical report, VRVis Research Center, www.VRVis.at, Feb. 2002. TR-VRVis-2002-003.

[5] B. Jobard and W. Lefer. Creating evenly spaced streamlines of arbitrary density. In *Visualization in Scentific Computing '97*, 1997.

[6] B. Jobard and W. Lefer. Multiresolution flow visualization. In *WSCG 2001 Conference Proceedings*, 2001.

[7] R. S. Laramee, H. Hauser, H. Doleisch, F. H. Post, B. Vrolijk, and D. Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23(2):203–221, June 2004.

[8] L. Li and H.-W. Shen. Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(3), 2007.

[9] O. Mattausch, T. Theubl, H. Hauser, and M. E. Groller. Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In *Proc. of the 19th Spring Conference on Computer Graphics*, pages 213–222, 2003.

[10] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *Visualization '05*, pages 479–486, Oct. 2005.

[11] G. M. Nielson. Radial hermite operators for scattered point cloud data with normal vectors and applications to implicitizing polygon mesh surfaces for generalized CSG operations and smoothing. In *IEEE Visualization*, pages 203–210, 2004.

[12] M. Schlemmer, I. Hotz, B. Hamann, F. Morr, and H. Hagen. Priority streamlines: A context-based visualization of flow fields. In *Eurographics/IEEE-VGTC Symposium on Visualization*, 2007.

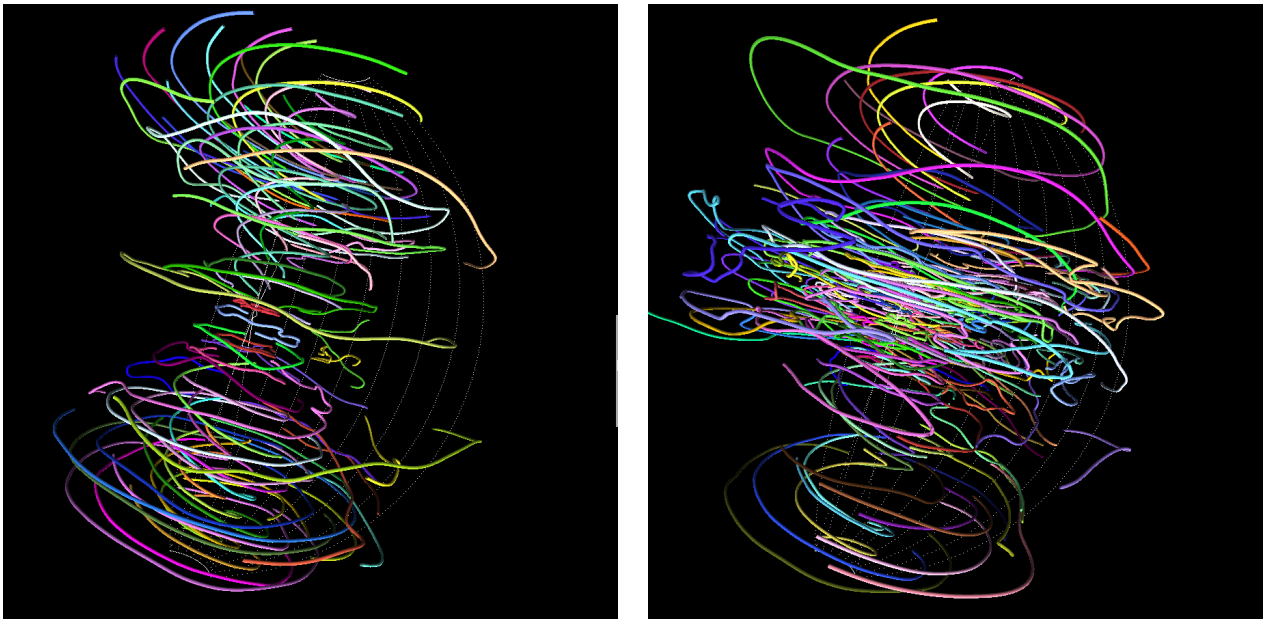
[13] G. Turk and D. Banks. Image-guided streamline placement. In *Proc. 23rd Ann. Conf. Computer Graphics and Interactive Techniques(SIGGRAPH '96*, pages 453–460, 1996.

[14] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Visualization '00*, pages 163–170, 2000.

[15] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3d streamlines. In *Visualization '05*, pages 471–478, Oct. 2005.

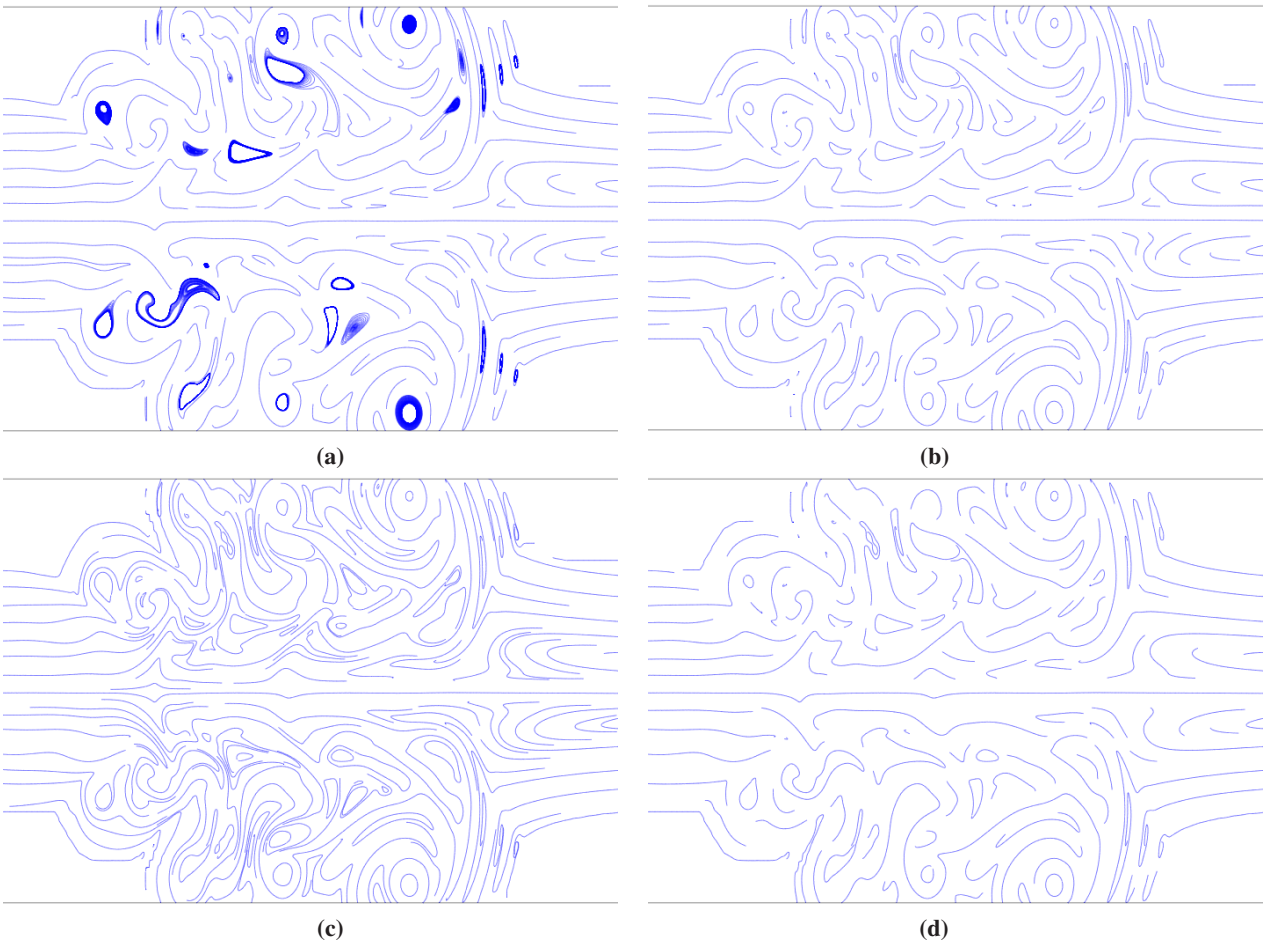
[16] R. J. M. I. Zhanping Liu and J. Groner. An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):965–973, Sept. 2006.

This work was performed under the auspices of the U. S. DOE by UC, LLNL under Contract W-7405-Eng-48.



(a) **(b)**

Fig. 13. Relativistic MHD simulation magnetic field data using variable density functions emphasizing: **(a)** Polar axis region **(b)** Equatorial region



(a) **(b)**

(c) **(d)**

Fig. 14. Swirling jet data set. **(a)** Euclidean metric, no self-separation testing. **(b)** Euclidean metric with self-separation testing and closure. **(c)** Similarity metric, same number of streamlines as (b). **(d)** Euclidean metric, same separation threshold as (c). The separation threshold is 3.0% of the field width for (a), (b) and 3.5% for (c) and (d). In (c), the window size is 10.0% of the field width and the shape coefficient α is 3.0.